Durham E-Theses

Algorithms and architectures for the multirate additive synthesis of musical tones

Phillips, Desmond Keith

Algorithms and Architectures for the

Multirate Additive Synthesis of Musical Tones

by

Desmond Keith Phillips

A doctoral thesis submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy

School of Engineering, Durham University, UK

December 1996

Abstract

In classical Additive Synthesis (AS), the output signal is the sum of a large number of independently controllable sinusoidal partials. The advantages of AS for music synthesis are well known as is the high computational cost. This thesis is concerned with the computational optimisation of AS by multirate DSP techniques. In note-based music synthesis, the expected bounds of the frequency trajectory of each partial in a finite lifecycle tone determine critical time-invariant partial-specific sample rates which are lower than the conventional rate (in excess of 40kHz) resulting in computational savings.

Scheduling and interpolation (to suppress quantisation noise) for many sample rates is required, leading to the concept of Multirate Additive Synthesis (MAS) where these overheads are minimised by synthesis filterbanks which quantise the set of available sample rates. Alternative AS optimisations are also appraised. It is shown that a hierarchical interpretation of the QMF filterbank preserves AS generality and permits efficient context-specific adaptation of computation to required note dynamics. Practical QMF implementation and the modifications necessary for MAS are discussed. QMF transition widths can be logically excluded from the MAS paradigm, at a cost. Therefore a novel filterbank is evaluated where transition widths are physically excluded.

Benchmarking of a hypothetical orchestral synthesis application provides a tentative quantitative analysis of the performance improvement of MAS over AS. The mapping of MAS into VLSI is opened by a review of sine computation techniques. Then the functional specification and high-level design of a conceptual MAS Coprocessor (MASC) is developed which functions with high autonomy in a loosely-coupled master-slave configuration with a Host CPU which executes filterbanks in software. Standard hardware optimisation techniques are used, such as pipelining, based upon the principle of an application-specific memory hierarchy which maximises MASC throughput.

Acknowledgements

Blessed Cecilia, appear in visions
To all musicians, appear and inspire
Translated Daughter, come down and startle
Composing mortals with immortal fire.

- W. H. Auden, "Song for St. Cecilia's Day", 1940

Contents

Figures

Tables

I confirm that the thesis conforms with the prescribed word length for the degree for which I am submitting it for examination.

I confirm that no part of the material offered has previously been submitted by me for a degree in this or any other University. If material has been generated through joint work, my independent contribution has been clearly indicated. In all other cases material from the work of others has been acknowledged and quotations and paraphrases suitably indicated.

<div align="center">

Signed: ....

Dated: ....

</div>

1. Introduction

1.1 A Tutorial on 'Classical' Additive Synthesis

1.1.1 Definition and Properties

$$y[n] = \sum_{i=1}^{S} A_i[n]\sin(\Phi_i[n]) \text{ with } \Phi_i[n] = \Phi_i[n-1] + \frac{2\pi}{f_s}F_i[n] + \phi_i[n] - \phi_i[n-1] \quad (1.1)$$

$A_i[n]$, $F_i[n]$, $\phi_i[n]$ = amplitude, frequency and phase envelopes of ith sinusoid

f_s = industry - standard digital audio sample rate e.g. CD at 44.1kHz

n = sample index

The digital form of Additive Synthesis (AS) as defined in equation (1.1) is a formulation of the Inverse Fourier Transform (IFT) where the superposition of S discrete-time sinusoidal oscillators in time-domain maps to a spectrum of S discrete lines in frequency domain (De Poli, 1983). Given a sufficiently high value of S and independent control over each oscillator (in terms of frequency, amplitude and phase envelopes), signals with a rich time-evolving spectrum can be synthesised. A particular application is the synthesis of musical tones which can often be expressed as a sum of harmonically related sinusoids or "partials". Each partial is mapped to an oscillator and the spectrum of the tone is quantified by two vectors of length S (for frequency and amplitude). These determine the 'spectral envelope' of the signal and give precise control over the quality of timbre perceived by the listener: a phenomenon first observed by Helmholtz (1863). The application of AS to music can be summarised by four characteristic properties:-

- Simplicity. AS is completely defined by the 'one-line' eqn. (1.1) and is therefore compatible with a VLSI implementation philosophy (Houghton et al, 1995).

- Analysis support via the Fourier transform to enable resynthesis of acoustic sources. Realistic reproduction of familiar sounds is an indispensable feature of a successful music synthesis algorithm (Risset and Mathews, 1969).

- Generality; it can model the spectral evolution of arbitrary sounds, unifying resynthesis of acoustic sources and "conceptual" sound generated from artificial control data. Traditionally, these domains have been logically distinct e.g. sampling and FM synthesis (Smith, 1991).

- Transformations on spectral data have an intuitive relationship with the listener's perception. Common operations are more logical to express in frequency than time domain (e.g. pitch shifting, time stretching). Combined with the algorithm's generality, expressive customisable control interfaces are facilitated (White, 1995).

1.1.2 Disadvantages

AS has a high computational cost in its native form of eqn. (1.1) because of (i) the overheads of execution itself and (ii), more critically, the unresolved problem of how control information is generated in the first place (Rodet and Depalle, 1992). Two streams of envelope data ($A_i[n]$ and $F_i[n]$) are incident upon each oscillator, each of which forms but a small additive energy component in the final audio signal $y[n]$: dynamic phase enveloping via $\phi_i[n]$ is usually omitted if inter-partial phases are correctly initialised and harmonicity is assumed (Horner and Cheung, 1995). A useful analogy for AS is that of data-compression where the incident control bandwidth far exceeds that of the output. A contrast is drawn with classical parametric synthesis techniques (e.g. FM synthesis - section 2.2.3) which have the opposite form. High control bandwidth is attributable to the combination of the three following factors:

- Envelopes should be updated, in ideal circumstances, at f_s to permit the accurate synthesis of transient detail and avoid the appearance of modulation sidebands in the spectrum caused by coarse time quantisation in envelopes (Jansen, 1991).

- An instrument voice will typically require >20 partials for satisfactory resynthesis (many higher harmonics lie below the perceptual threshold and can be omitted) (Moorer, 1977). The lower the note, the more are required: 80 partials is not an untypical requirement for bass notes with a significant treble component. (Haken, 1991)

- To synthesise an ensemble, one should ideally superimpose an instantiation of each sounding note: in practice, the slight detuning of actual instruments generates a rich texture (Meyer, 1993). In contrast, sampling is more economic but offers little opportunity for expressive control (see section 2.2.1.) (Smith, 1991).

Hence a 100-note ensemble with, say, 40 oscillators per note will require a net control rate of $100 \times 40 \times 2 \times f_s = 352.8 \times 10^6$ samples per second assuming f_s=44.1kHz. Using 16-bit envelope samples, this bandwidth represents about one CD's worth of data per second (600MBytes). Clearly, this is outside the realm of low-cost microcomputer implementation in the present-day which is our desired goal. A more typical supercomputer-based strategy is exemplified by Kaper et al (1995).

1.1.3 Piecewise Linear Envelope (PWL) Modelling of Envelopes
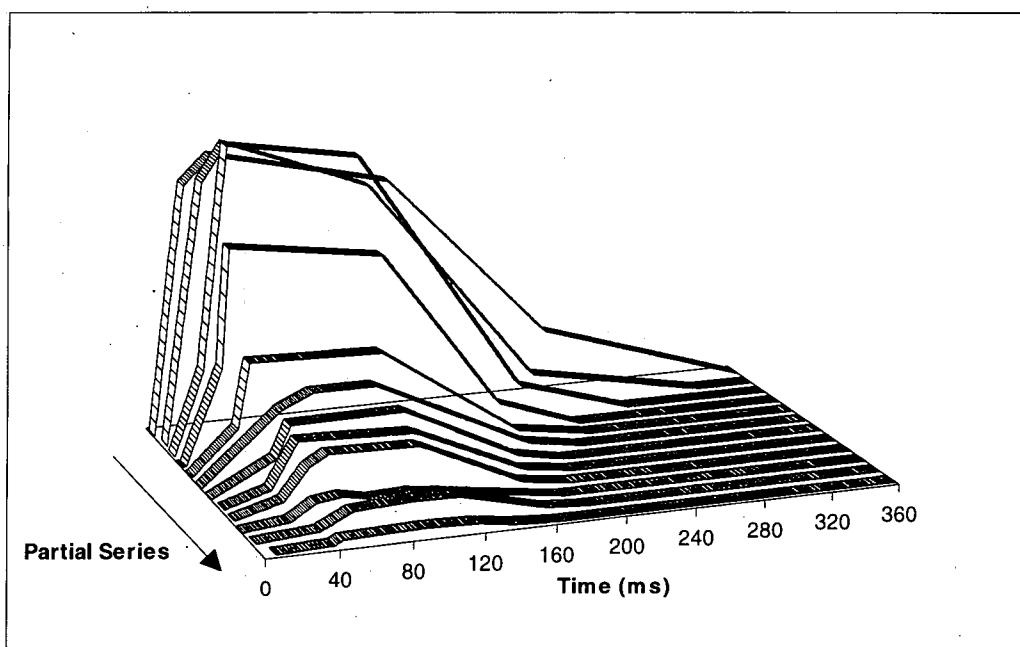
Figure 1.1   PWL Approximation of a Trumpet Note

A significant reduction in control bandwidth is accomplished by using piecewise linear (PWL) models of raw envelope data extracted from a Fourier analysis of acoustic sources (see section 2.3). Such envelopes are characterised by jitter superimposed upon a smooth profile. The latter is preserved as a compact set of breakpoints coded as (time, height) by PWL modelling once jitter is smoothed out (Moore, 1990). Grey (1977)

demonstrated via listening tests that resynthesis from a PWL model is musically indistinguishable from the original indicating that envelope jitter is perceptually superfluous. Fig. 1.1 illustrates the PWL amplitude envelope set for the first twelve partials of a complete trumpet sample; an often-cited example of Grey's work which uses 176 breakpoints to model $A_i[n]$ and $F_i[n]$ for $1 \leq i \leq 12$ (Moorer et al, 1978). The data set is thus 352 samples as compared to 360ms $\times f_s$ (44.1kHz) = 15876 samples, or about 45:1 compression. Practical compression factors of 100:1 are reported elsewhere by Serra and Smith (1990). To extend the analogy of section 1.1.2, a static PWL frequency-domain representation of breakpoints is uncompressed into a set of envelope time-series, representing the time-evolving spectrum, and re-compressed by AS into a time-domain image.

The technique is flexible in that, if desired, high-resolution detail such as high frequency transients may be encoded with greater precision and a corresponding penalty in efficiency. Transformation of compressed envelopes is computationally efficient because only breakpoints require manipulation, and intuitive because the 'topographical' representation of a time-evolving spectrum is retained. Uncompression in real-time is simplified if breakpoints are recoded as (*gradient, height*) as only discrete integration and a breakpoint detection mechanism are necessary for envelope synthesis (Snell, 1977). PWL modelling is a milestone in the development of AS as a practical tool because it reduces the control data problem from an intractable level to a highly manageable one, without a significant diminution of the properties set out in section 1.1.1.

1.2 On the Specification of an Ideal Oscillator Bank

The properties required of a successful AS implementation may be divided usefully into (i) performance criteria which should be satisfied as far as possible and (ii) concrete design constraints which must be satisfied.

1.2.1 Performance Criteria

- The unit cost of an oscillator should be as low as possible (calculated by cost of oscillator bank / S) to facilitate an affordable additive synthesiser with high S.

- A sufficient number of output streams should be available for spatialisation in the construction of the final multichannel sound image.

- Functional Transparency. Implementation details should not seriously compromise the advantages of AS as set out in section 1.1.1.

1.2.2 Design Constraints

- The oscillator bank output satisfies an industry-standard digital audio format, e.g. 16-bits at f_s=44.1kHz (CD), to ensure high fidelity synthesis.

- Support for linear interpolation between the breakpoints of PWL envelopes for $A_i[n]$ and $F_i[n]$ at a resolution of f_s for high fidelity PWL-AS.

- Real-time system latency from the reception of an event (e.g. MIDI note on) to signal output should be no greater than $T_{max} \cong 10^{-2}$s for imperceptibility.

1.3 Conventional Solutions to AS Computation

1.3.1 The Analogue Background to Digital AS

To understand the importance of AS to music today, it is useful to trace briefly its ancestry. Its origins lie in Pythagoras' theory, derived from experiments with string lengths in harps, that harmony and consonance in music are related to frequency ratios of the numbers 1,2,3 & 4 which is explained physically by the interaction of harmonically related partials (Taylor, 1965). Then, in 1822 whilst working on the physics of heat flow, Fourier developed his theorem and laid the foundations for mathematical signal analysis. Arbitrary periodic functions could be explained in terms of a 'Fourier' series of harmonically related sinusoids at individual phases and amplitudes (Oppenheim, 1983). These ideas were taken up by Helmholtz (1863) as they corresponded with his practical investigations into the physics and perception of sound. He was able to demonstrate that the quality of musical tones was dependent on the spectral envelope of its partial series. Helmholtz also built one of the first additive synthesisers using an electrically driven tuning-fork for each partial with an adjustable acoustic resonator to regulate amplitude.

The telephone, invented by Bell in 1876, represented sound as a varying electrical current and enabled the construction of electromechanical additive synthesisers. Early prototypes involved substantial engineering. Cahill's telharmonium used gearing to establish harmonic ratios and alternators to generate waveforms which were mixed by a resistor network and transmitted direct to earpieces at subscribers' households. Shafts up to 30 feet in length were necessary to carry alternators of sufficient power (Manning, 1985). Subsequently, invention of the thermionic valve and loudspeaker permitted amplification of audio signals and consequently electromechanical tone generators could be miniaturised and manufactured cheaply enough to be put into commercial products. Two famous examples are the Hammond (electromagnetic) and the Compton Electrone pipeless organs (electrostatic), both creating complex waveforms by controlled mixing of harmonically related 'primitive' waveforms (Comerford, 1993).

The introduction of semiconductor and digital computer technology (late 1940's) made many new synthesis techniques possible. AS using rotating tone wheels was superseded first by analogue (1960's/70's) and then digital waveform generation (1980's). However, the concept was adopted by computer music researchers because it is a mathematically rigorous and general way to theorize about music though expensive to compute. As a result of the research effort of the last thirty years, a better perspective has emerged. AS is seen as the foundation layer of a wider 'spectral modelling' paradigm that allows composers to work intuitively with sounds in frequency domain (Smith, 1991). The development of digital AS, as opposed to analogue, is documented in section 2.3: 'AS' and 'digital AS' are interchangeable terms nowadays.

1.3.2 The Traditional Oscillator Bank

A 'direct-form' implementation of AS differs little from the electromechanical schemes outlined. All that is needed is a bank of S identical, independently controllable sine oscillators. More usually, given the higher relative clock rates of digital integrated circuits compared to sample rates for digital audio ($>10^7$Hz versus $<10^4$Hz), a single sine oscillator multiplexed S times suffices. It is expressed in the previous 'c' code loop which includes the discrete integration required for PWL envelope uncompression (omitting

breakpoint detection for clarity). S iterations occur every sample period, accumulated to form a single sample op.

```
op=0;
for (n=0;n<S;n++)    {
        amp_acc[n]+=amp_inc[n];
        freq_acc[n]+=freq_inc[n];
        phase_acc[n]+=freq_acc[n];
        if (phase_acc[n]>PI)
        phase_acc[n]-=2*PI;
        op+=am_acc[n]*sin(phase_acc[n]);
}
```
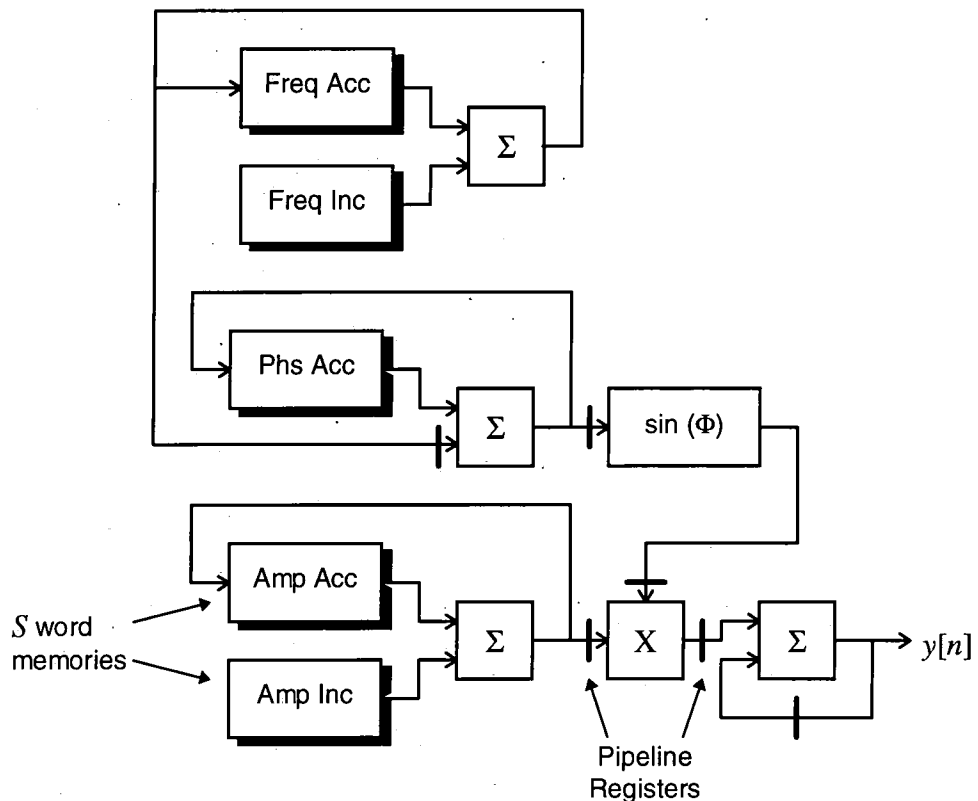

Figure 1.2   Traditional Oscillator Bank

In this form, the mapping to a dedicated architecture, as illustrated schematically in Fig. 1.2, is trivial. The data dependencies between functional blocks are first order and hence pipeline registers may be included as indicated; a *systolic* representation of dataflow within the AS algorithm (Leiss, 1995). The depth of logic and hence, propagation delay, between registers is minimised so that the clock speed, and value of S can be maximised. The chief bottleneck is the computation of $\sin(\Phi_i[n])$ which is a non-linear

transformation and often performed by a Look-up Table (LUT). Concurrency in both computation and state memory access is fully exploited. In contrast, execution of the code loop on a standard scalar microprocessor would result in a memory bottleneck for state variable access (Freed et al, 1993).

Within the terminology of parallel computing, AS is a *fine-grain data-parallel algorithm*. Fine-granularity is due to the low process complexity; one iteration of the code loop, in contrast to a large process set size, S. It is data-parallel because of the incidence of many independent streams of control information (in the PWL form) upon parallel instantiations of the same process. With a strongly classifiable algorithm, the most effective implementation is often a dedicated architecture. It is a philosophy of "form follows function". Positive evidence is the fact that the systolic model has survived several generations of research prototypes as exemplified by Snell (1977), Jansen (1991) and Houghton et al (1995). A convenient label for this systolic architectural form is the Traditional Oscillator Bank (TOB).
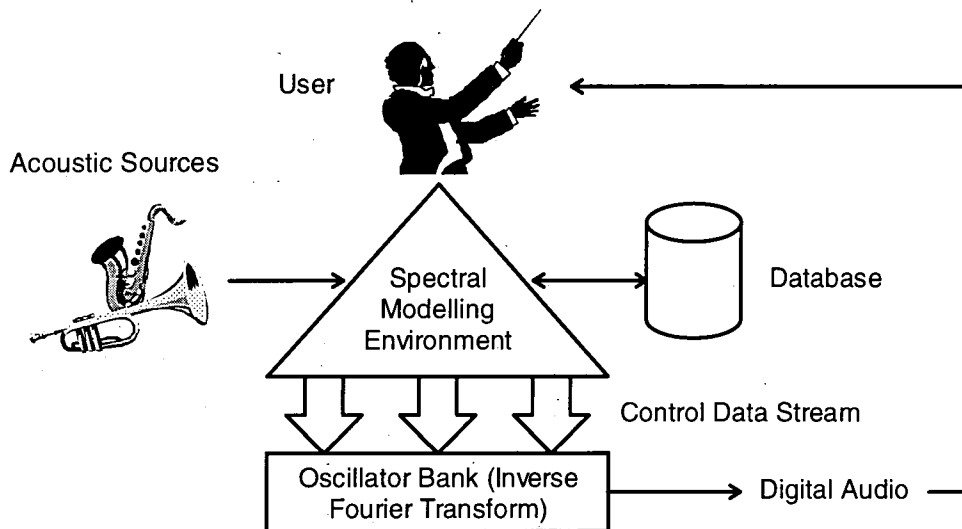
Figure 1.3   The Role of an Oscillator Bank in Spectral Modelling

To adopt a contrasting 'top-down' perspective consider Fig. 1.3, an overview of the complete Spectral Modelling Environment (SME). A recent example is 'Lemur' (Fitz and Haken, 1995). The reader is also referred to Osaka (1995), Freed (1995), Hill (1991) and Feiten and Ungvary (1990). Two distinct systems can be identified; (i) modelling and (ii) synthesis. Modelling uses high levels of data abstraction and is a natural application

for software. Powerful CPU's permit sophisticated object-oriented, graphical SME's. Towards the base of this pyramidal hierarchy, parallel streams of control data are generated in real-time. An oscillator bank is necessary to perform the IFT represented by AS and should have sufficient synthesis resources to underpin the higher levels of the hierarchy. In contrast to modelling, an oscillator bank is a deterministic process and a natural application of custom hardware, because of the absence of an instruction stream. Reiterating the point of section 1.1.2, functionality is characterised by data *transformation* rather than data processing.

1.3.3 Impact of New Technology

The issues of section 1.3.2. must be set in the context of reductions in VLSI device geometry, permitting smaller faster devices. As a result, both clock rates and packing density have risen giving a net CPU performance increase of about 40% per annum, sustained for the last twenty years. Many synthesis algorithms can now be implemented in software in parallel in real-time on a single high-performance CPU, including forms of AS (Smith, 1991). The expense and lead-time of developing dedicated hardware is no longer understood to be necessary because of innovations in 'software' synthesis algorithms which can be ported to newer faster platforms, provided that they share a common operating system such as UNIX. Smith (1991) justifies this bias towards 'software' synthesis.

> *"Another problem with supporting special-purpose, computer-music hardware is that it can be obsolete by the time its controlling software is considered useable"*

As explained in Chapter 2, physical and spectral modelling are, in conjunction, two of the most active areas within music technology research in the 1990's because technological advance is making the implementation of their computationally intensive algorithms (including AS) cheaper and thus available to a wider public. The conceptual framework of modelling provides the necessary "degrees of freedom" required by creative artists. In contrast, algorithm-oriented approaches (see section 2.2.3), typical of the first generation of digital synthesisers in the 1980's, are falling into increasing disfavour: at that time, economy of implementation was the primary objective and resulted in restricted sound

palettes with idiosyncratic control mechanisms. At present, spectral modelling lags its physical counterpart in commercial exploitation due to the problem of high control bandwidth - but its long-term future is secure for the reasons set out in section 1.1.1. Smith (1991) makes an unambiguous prediction:-

> *"It is anticipated that synthesis in the future will be dominated by spectral and physical modelling."... "Spectral modelling is the more general of the two..."*

There is, therefore, both a motive and means to develop products using AS because the commercial potential is large. In deference to the philosophy discussed previously, researchers are looking at AS from a software synthesis perspective. For instance, an alternative to an oscillator bank is the Inverse Fast Fourier Transform (IFFT), the most efficient way to transform a spectrum into a waveform (as discussed in section 2.4). It is a block transform and an outer kernel of pre/post-processing is necessary for effective simulation of the sample-level execution of equation (1.1). Its complexity leads more naturally to a software rather than hardware implementation, but executing on a general-purpose CPU it offers performance competitive to a TOB and is much cheaper to build: partly because 'flagship' CPUs developed by major companies run at the highest clock rates of all (Freed et al, 1993). Therefore, interest has declined in refining dedicated synthesis architectures. Serra (1994) summarises this attitude:

> *"An additive synthesis implementation based on the IFFT has been proposed which is more efficient than the traditional oscillator bank..."*

1.4 Thesis Research Direction

1.4.1 Optimisation of Sample Rate in Note-Based AS

$$c_{as} = aSf_s \qquad (1.2)$$

The substance of this thesis is to investigate how far criticism, as expressed in section 1.3.3, about TOB inefficiency is justified given the strength of the arguments in its favour as set out in section 1.3.2. An intial step is to observe that the cost c_{as} of computing AS

via equation (1.1) is given by equation (1.2) where a is the cost of a single oscillator update. PWL modelling is an economic SME representation but, for conversion into a time-domain signal for playback, uncompression is required in real time and hence benefits of the technique fall outside the scope of eqn. (1.2). Control data bandwidths of the order discussed in section 1.1.2 recur and the implications of equation (1.2) cannot be avoided, unless a software IFFT solution is used. To reduce c_{as} there are three avenues of promise;

1. Reducing a by low-level optimisations to oscillator bank hardware

2. Reducing S by selective removal of partials from the synthesis set

3. Reducing the (mean) oscillator sample rate f_s.

This thesis concentrates upon developing an algorithm to reduce f_s and an architecture to reduce a, leaving S as unconstrained as possible so that the subjective qualities (e.g. sound richness) of AS are unimpaired. Optimisation of a is mature after a protracted evolution of TOB prototypes, but it is asserted in Chapters 8 and 9 that higher level refinements in synchronisation and scheduling are still possible. Minimising S is already practiced in that only the highest energy partials are necessary for acceptable resynthesis quality: 'receiver coding' is a more sophisticated (and problematic) technique which is discussed in section 1.3. The reason for setting f_s to e.g. 44.1kHz is that the audio spectrum extends up to >20kHz and reducing f_s to e.g. 8kHz would produce 'telephone-quality' music of unacceptable fidelity. Hence this strategy is not an option.

Fortunately, this thesis proposes a more subtle approach for reducing f_s which is predicated on the universal acceptance of note as a fundamental concept in music. Ironically, computer-generated music is free from the physical laws that give rise to notes in acoustic instruments, and the universality of note is less valid for electroacoustic composers. However, the unifying factor in all music is that it is perceived by the human auditory system which has a well-defined anatomy and cognitive behaviour, studied through psychoacoustics. Holst (1963) elucidates that note-based music is founded on structures present in spoken language and that there is a phonetic, syntactic and semantic correlation between the two. Notes represent the lowest, phonetic level of musical language and have certain, definable properties, just as utterances in speech have a

symbolic notation drawn from a small set of phonemes which are determined by vocal tract anatomy.

The TOB is generalised for arbitrary digital audio frequency signals, rather than just note-based music. All oscillators have a default frequency ceiling, according to Nyquist's sampling theorem, of $f_s/2$ implying that, during their lifecycle, they require frequency dynamics over the entire audio range. This is an erroneous assumption for note-based music, explaining the discrepancy between the computational expense of AS on a TOB and its apparent efficiency of implementation as highlighted in section 1.3.3. In AS terms, notes are characterised by (1) a finite lifecycle of attack, decay, sustain and release and (2) an evolving pitched timbre definable as partial series with $F_i[n]$ and $A_i[n]$ envelopes. Superimposed upon this is "expression" in the form of dynamics, such as tremolo, vibrato or glissando. However, the concept of note remains an intrinsic and absolute determinant in musical discourse.

1.4.2 Determination of Optimal Sample Rates

The observation is made that at or before note onset, data is available on pitch, expected pitch modulation range, and the relationship of partial frequency ratios to modulated pitch (usually considered as constant) which enables the upper bound of $F_x[n]$ for each partial x to be estimated as $f_{max}(x) = \max(F_x[n])$. This data is an attribute of the voice to be synthesised and is therefore usually available *a priori* by default. Given that AS uses sinusoidal basis functions, Nyquist's sampling theorem may be re-interpreted to state that the optimum, or *critical*, time-invariant sample rate for alias-free synthesis of x is given by eqn. (1.3) where, generally, $f_{opt}(x) << f_s$. Not only is an oscillator's update rate reduced, but so also is the bandwidth of PWL envelope uncompression. Potential savings are indicated by the ratio $f_s / f_{opt}(x)$ and the repercussions are far-reaching.

$$f_{opt}(x) = 2f_{max}(x) \qquad\qquad (1.3)$$

For illustration, Fig. 1.4. plots the $F_i[n]$ envelope set ($1 \leq i \leq 12$) for Grey's trumpet as a counterpart to Fig. 1.1 (Moore, 1990). A rapid rise in partial frequencies is evident during note attack which stabilise at harmonic intervals during sustain, but at no point do

trajectories stray above 3.6kHz, which implies a maximum $f_{opt}(x)$=7.2kHz which is about

6:1 as efficient as synthesis at f_s=44.1kHz. Indeed, for lower frequencies such as the

fundamental at 300Hz, $f_{opt}(x)$=600Hz which implies an efficiency ratio of about 74:1 on

the basis of the previous example. A note has a theoretical series of harmonics extending

up to $\cong$20kHz. For instance, with f_x=27.5Hz (A0), 727 sinusoids are required posing a

formidable computational burden. However, the logarithmic frequency response of the

human ear and the general observation (Sandell, 1994) that the spectral envelope of

stationary musical timbres converges towards zero amplitude above 5kHz (disregarding

some notable exceptions like a muted trumpet) both indicate that low frequencies should

have greater priority for inclusion in the set of synthesised partials than high frequencies

in order to reflect their relative perceptual weighting ('receiver coding' is elaborated

upon in sections 1.3 and Chapter 6). Critical sampling therefore facilitates large savings

in AS throughput because computation of these low frequency partials at the non-critical

rate of f_s is highly inefficient.
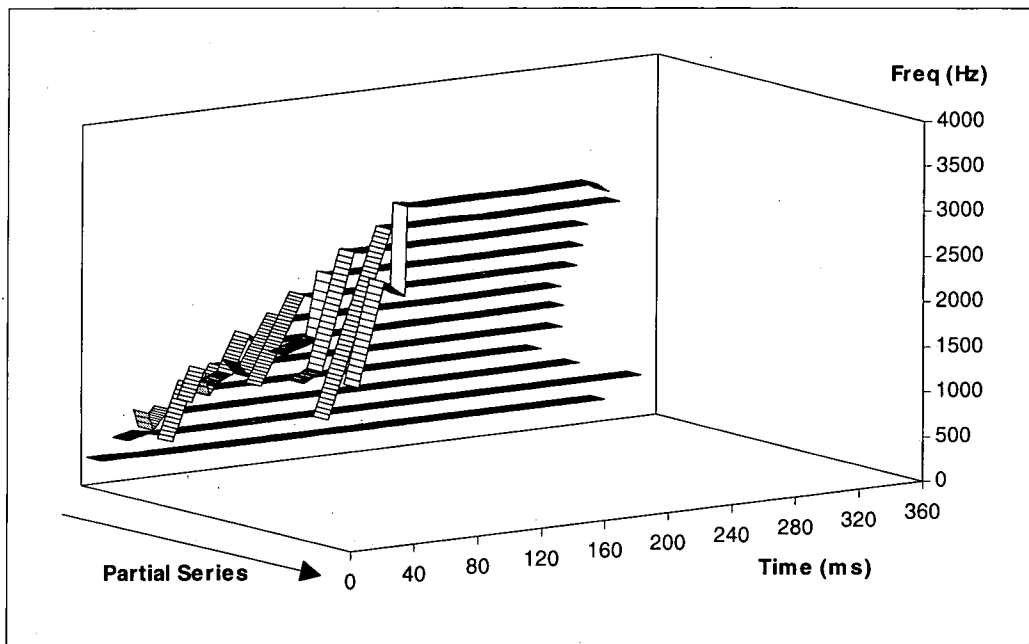
Figure 1.4   Frequency Envelopes for Grey's Trumpet

Just as $F_x[n]$ of x has an upper bound $f_{max}(x)$, so it has a lower bound of

$f_{min}(x)$=min($F_x[n]$). The interpretation of Nyquist's theorem can be extended to give eqn.

(1.4) if, conceptually, x is modulated up to $f_{min}(x)$ from a baseband spanning from DC to

$(f_{max}(x) - f_{min}(x))$. This results in a much lower $f_{opt}(x)$ because it is unusual for $F_x[n]$ to extend frequency modulation down to DC in music synthesis. Instead eqn. (1.4) optimises $f_{opt}(x)$ to the modulation range required by x irrespective of operating frequency resulting in large potential savings for high frequency partials with a narrow $F_x[n]$ modulation range. To generalise, for a note with $\alpha = max\ pitch\ /\ min\ pitch$ and pitch-invariant partial frequency ratios, $f_{min}(x) = f_{max}(x)/\alpha$. For instruments with a low pitch modulation (e.g. a piano), $\alpha \cong 1$ implying that $f_{opt}(x)$ will be low for all x. However, fundamental physical properties of musical signals mitigate against such a direct optimisation.

$$f_{opt}(x) = 2(f_{max}(x) - f_{min}(x)) \qquad (1.4)$$

The Heisenberg inequality, given in equation (1.5), states that the product of time (Δ_t) and frequency (Δ_f) resolution has an upper bound (Gabor, 1947), which can be interpreted in an AS context as the observation that envelope control rate has an inherent proportionality with oscillator operating frequency. Alternatively, the inequality becomes an approximate equality. High frequency partials, such as transients, possess features at a fine time resolution and require a rapid control rate. Conversely, low frequency partials vary relatively slowly with time and may exploit a lower time resolution. Such reasoning also forms the basis for the new discipline of 'wavelet' theory (Rioul and Vetterli, 1991). A frequency domain interpretation is that control rate is proportional to the width of modulation sidebands generated by $A_i[n]$ (less so by $F_i[n]$) giving a 'constant-Q' time-frequency paradigm (see section 3.1.4). For AS, the critical sampling of eqn.(1.3) encapsulates these principles whereas eqn. (1.4) does not.

$$\Delta_t \Delta_f \geq \frac{1}{4\pi} \qquad (1.5)$$

However, eqn. (1.4) is valid for the exception to the general rule when the left-hand side of eqn. (1.5) is much greater than the right-hand side as is the case when high frequency partials have a slow control rate and little transient character. For these reasons, it can be understood that eqn. (1.4) can be used to estimate $f_{opt}(x)$ if $\{f_{max}(x), f_{min}(x)\}$ is determined

- more generally - from an estimate of the bounds of the time-evolving spectral mainlobe of x under modulation of $A_i[n]$ and $F_i[n]$ such that aliasing of external components does not degrade perceived synthesis quality.

1.4.3 A Proposal for Multirate Additive Synthesis

Returning to the realities of VLSI development, as other architectures have fallen in and out of favour, the TOB form has remained immune - due to the *scalability* of AS and the fact that increasing clock rates merely increases S. It is a proven starting point for the work presented in this thesis. If each sinusoid x in an oscillator bank has an independent, optimal sample rate $f_{opt}(x)$, then many different sample rates must be supported simultaneously. Such a digital signal processing system is classified as "multirate" (Crochiere and Rabiner, 1983) and hence Multirate Additive Synthesis (MAS) is used as a label in later discussions. It is the purpose of this thesis to establish (i) a theoretical MAS paradigm, (ii) how it is subject to practical constraints, (iii) to derive some preliminary estimates about performance improvement as compared to classical AS and, finally, (iv) how MAS may be mapped into an efficient application-specific VLSI architecture.

The factors involved with determining $f_{opt}(x)$ in the context of note-based AS have been identified, but two further issues are raised which are pertinent to mention here. The first is that synthesis of x at $f_{opt}(x)$ creates sidebands around harmonics of $f_{opt}(x)$ which lie within the audio spectrum because $f_{opt}(x) < f_s$, representing audible time-quantisation noise and lending an unpleasant 'digital' quality to synthetic sound. Signal interpolation is required to raise the sample rate from $f_{opt}(x)$ to f_s to suppress this noise. Though an expensive process, interpolation can be applied to a sum of oscillators such that net savings in exploiting $f_{opt}(x)$ exceed the overhead. The second issue is one of multirate oscillator scheduling. The TOB executing at a single rate of f_s requires only a simple 'round-robin' schedule. Supporting S unique incommensurate $f_{opt}(x)$'s in real-time poses a considerable challenge and quantisation to a finite set of schedulable $f_{opt}(x)$'s is necessary.

1.4.4 Thesis Structure

In Chapter 2, the context and development of AS are reviewed. Firstly, there is a brief summary of alternative synthesis techniques which highlights the excellence of AS as outlined in section 1.1.1. Secondly, the development of digital AS and spectral modelling is traced building upon section 1.3.1. A literature survey of proposed AS optimisation methods is presented which stresses the importance of minimising computation without reducing AS generality. Thirdly, the use of the IFFT in emulating a TOB is documented in detail because it is emerging as one of the most popular and economic methods for implementing AS to which MAS offers an alternative solution.

A paradigm for MAS is developed in Chapter 3 from first principles. After a brief review of the basic operations of multirate DSP in the context of the sinusoidal synthesis, it is demonstrated how AS is transformed into MAS by the introduction of multirate filterbanks in time-domain that relate to specific subband decompositions in frequency domain. The application of various decompositions to expected frequency distributions of partials in note-based music is discussed and concludes with a cost / benefit analysis of MAS. Arising from this, a paradigm based on QMF filterbanks is proposed which uses a hierarchical subband decomposition.

Practical details of QMF filterbank implementation are discussed in Chapter 4. Both FIR and IIR designs are possible and each has attendant advantages and disadvantages. For functional transparency in MAS, a number normalising modifications to the QMF structure are specified. In particular, the phase-distortion inherent in QMF filtration is of concern. Chapter 5 identifies solutions to the problem of non-infinitesimal transition widths in QMF filterbanks. A method for their logical exclusion is presented and discussed. In the light of the shortcomings of QMF filterbanks, a MAS-specific design is presented - the Physical Exclusion Filterbank (PEF) - and supported by simulation results. Finally, the range of filterbank options, and their application to MAS, are summarised.

To reach some speculative conclusions about the performance improvement afforded by MAS, Chapter 6 profiles an experimental simulation of the resource allocation of a hypothetical, yet feasible large-scale practical MAS application: the synthesis of a

symphony orchestra with one AS voice per instrument. The experiment involves real musical scores and instrument timbres. The resulting allocation histograms are applied to models that provide benchmarks of the speedup and cost of MAS for a given filterbank configuration. Another important aspect of the chapter is that the experimental scenario provides a context for discussing many of the higher-level, application-oriented issues of MAS in contrast to its low-level computation.

A key factor in the design of a successful MAS implementation is an efficient multirate sinusoidal oscillator bank. Digital sinusoid generation is a relatively mature subject and thus a significant proportion of Chapter 7 is devoted to a literature review of current techniques, with a bias on their efficacy for AS / MAS. Oscillator designs are classified under two headings as; (i) recursive and (ii) phase-accumulator based. The use of the CORDIC algorithm for complex sinusoid generation is investigated and results are presented. However, it is evidenced that a modification to the traditional look-up table (LUT) approach has best performance in terms of throughput / silicon area.

In Chapter 8, a systems analysis of a hypothetical MAS synthesiser indicates the utility of mapping MAS into an ASIC-based coprocessor (MASC) which forms part of a multiprocessor with general purpose CPUs executing software processes. Standard computer engineering concepts are applied in a novel context. The installation of a MASC within a multiprocessor topology is discussed and leads to the proposal for inter-processor communication via shared memory, which isolates high control bandwidths from the software processes via a MAS-specific memory hierarchy. When combined with an object-oriented data structure design, the fundamental basis of MASC thoughput optimisation by interleaved burst processing is acheived.

Chapter 9 completes the MASC functional specification by specifying all of the remaining higher-level scheduling and synchronisation algorithms and data structures that are necessary for MASC operation in the context of a multiprocessor MAS synthesiser. A quantitative review of expected MASC performance as a function of its design parameters (and in the light of Chapter 6) shows that MAS optimisation is manifest in the MASC by a reduction in the required internal clock rate. Finally, a data-flow model

of the MASC and its major functional units is provided, leaving the way open for subsequent behavioural simulation and prototype implementation.

1.4.5 Publications Arising from Thesis Research

- Conference Publications:-   (Phillips et al, 1994), (Phillips et al, 1996a)

- Journal Publications:-   (Phillips et al, 1996b)

2. Additive Synthesis: Context and Development

2.1 Overview

AS is but one of many music synthesis algorithms. Smith (1991) suggests a taxonomy of four categories; (i) Processed Recording, (ii) Spectral Modelling, (iii) Physical Modelling and (iv) Abstract Algorithm. Section 2.4 is devoted to tracing the development of (ii) as it is of particular relevance to this thesis. Section 2.2 is a review of the remaining categories with a primary motive of investigating how they compare with AS in both quantitative and qualitative terms (e.g. computational efficiency, intuition of control). The most significant advance that is identified is the application of the IFFT for TOB simulation. As MAS shares the same objective, this topic is documented in detail in section 2.4.

2.2 Alternative Synthesis Techniques

2.2.1 Processed Recording

'Processed Recording' is synthesis by the time-domain processing of recorded sounds. 'Sampling' constitutes a mere playback of the original sound. There are no implicit control parameters, they must be contrived. Pitch is altered by varying the playback sample rate, and timbre is shaped by applying time-varying filtration. Sustaining notes are simulated by looping the sample. All of these operations introduce perceptible distortions away from the original sound. An alternative is to compile a library of samples that encapsulate the acoustic source's timbral range as a function of its control parameters. However, storing many complete variants of the same basic sound is an inefficient use of memory. Notwithstanding, sampling is a most effective technique where the acoustic source has a minimum of performance control parameters, such as percussion instruments, where a single recording may suffice (Smith, 1991).

'Wavetable' synthesis is a logical development of the looping principle: only one period of the tone is stored eliminating much of the memory redundancy of sampling. Greater storage efficiency facilitates the compilation of a library of wavetables that more

completely characterizes an instrument's timbre range: a recent case is provided by Horner et al (1993). As the timbre of a note is a function of its pitch and evolves with time, an infinite number of tables are required, in theory, to describe this two-dimensional mapping. However, the process is generally smooth and therefore an efficient data-reduction technique is a quantised grid with intermediate timbres generated by linear interpolation between 'nearest neighbour' wavetables (partial phases must be normalized). Quantisation of the pitch and time axes should be as coarse as perceptual constraints will allow in order to minimise redundancy. A suitable example is acoustic piano resynthesis: the similarity of timbre between adjacent notes implies that quantising the pitch axis to individual notes creates redundancy. However, octave-level quantisation, say, is impractical because the fixed-length of wavetable implied at each pitch value has a bandlimited range of operation because (i) they are prone to quantisation noise at lower fundamental frequencies (an oversampled table is a solution) and (ii) higher harmonics may alias at higher fundamental frequencies. Therefore a compromise is often achieved. An advantage of the resulting restricted range of pitch modulation is that the complexity of voice-specific sample rates (c.f. sampling) can be avoided using a phase-accumulator approach at the expense of phase-jitter noise (see section 7.4.2). Mauchly and Charpentier (1987) discuss a typical hardware implementation.

For instruments with a number of interdependent control parameters e.g. a violin with three parameters of pitch, bowing velocity and distance from bridge, linear interpolation becomes multi-dimensional. This is called 'vector synthesis' (Smith, 1991) as the control parameters represent interpolated timbre as a vector in a n-dimensional bounded timbre-space (Wessel, 1979). For n parameters a minimum of 2^n control points are needed, representing vertices of the bounding hypercube, which can consume a substantial amount of memory space and bandwidth in implementation. It also makes the assumption that with all other parameters fixed, timbre varies linearly between the extreme limits of a single parameter. For timbre as a function of frequency over 7-octaves, this is likely to be unsatisfactory. Non-linearity is supported by PWL subdivision of the hypercube into a lattice (Haken, 1991). However, if the parameters are mutually independent, then only $2n$ wavetables are needed. For instance, FOF synthesis exploits the independence of pitch

and formant envelope in singing voice synthesis with n wavetables for n formants (Rodet, 1984). Formant envelopes can also be implemented by filter structures (Sandler, 1989).

Wavetables with interpolation represent an attempt to resynthesise the recorded instrument from a small data set. The analysis procedure is one of data-compression rather than the derivation of useful abstractions such as a spectral or physical model. However, the stored waveform approach is flexible, it can create purely synthetic sounds, generated 'off-line'. It is also possible to create impressive and original musical textures with deliberately intensive processing via 'granular' synthesis (Roads, 1978). Of relevance to AS is the fact that some systems have supported its basic principles by manipulating spectra that are written into wavetables via an IFT at run-time (Serra et al, 1990). It is not a full implementation of AS because independent control of partials is lost by their coalescing - for instance, exact harmonicity is assumed. However, the spectral definition is more compact and intuitive than an oversampled wavetable, and makes interpolation easier by default normalisation of partial phase (Smith, 1991).

2.2.2 Physical Modelling

Physical modelling synthesis uses discrete-time mathematical models that describe the physical behaviour of acoustic instruments, which without exception, have a form which includes excitation of a resonant system that possesses control parameters that modify resonant behaviour. Control interfacing mimics the typical interaction of player and instrument. With sophisticated control transducers, real-time physical models become as playable as their acoustic equivalents, giving rise to their classification as 'virtual' instruments. They also encapsulate naturally occurring phenomena such as non-linear and chaotic behaviour which is difficult to map into an inherently linear wavetable model of synthesis: an example is the octave jumps in pitch caused by overblowing on a clarinet. Also, a physical model does not express its timbre space or control mechanism in terms of a static lexicon of stored waveforms, rather it is implicit in the topology of the model, eliminating much potential for redundancy (Smith, 1991).

The most popular building-block for a physical model is the digital waveguide: a delay line that models a wave (pressure or displacement) travelling through a unit length of a one-dimensional medium, such as a taut string. Actually, two delay-lines operating in

opposing directions are needed to simulate bi-directional propagation. The Karplus and Strong (1983) 'plucked string' synthesis algorithm was first to indicate the potential of waveguide techniques. The output of a delay-line, initially filled with noise is fed back to the input via a low-pass filter. High frequencies in the initial noise burst decay rapidly leaving a lingering fundamental, determined by the waveguide length. Many other physical models of acoustic instruments dependent on a single resonator have been constructed successfully based on single wave-guides, such as woodwind instruments (Smith, 1987).

For instruments that are composed of interacting resonators, a more complex approach is needed. An example is simulation of the interaction between the forced vibration of a violin string and the passive resonance of the body. 'Waveguide' synthesis uses two-dimensional 'meshes' constructed with grids of waveguides connected by lossless scattering junctions, which simulate membrane stiffness and provide signal dispersion. Meshes can take the forms of plates and cylinders etc., the topology following the form of system to be simulated (Van Duyne and Smith, 1994). Examples of other constructs in the technology are non-linear reflections at boundaries that shape timbral evolution, and digital hammers that model all of the physical interaction required to simulate natural attack transients (Van Duyne et al, 1994).

The limitation of physical modelling is that one is fundamentally limited to resonant systems and excitation functions, and this is but a subset of musical sounds. Virtual instruments do open new possibilities for performer/instrument interaction but the emphasis on physical control creates a dependence on transducer technology: a conventional MIDI keyboard is an inadequate way to control a woodwind model. Timbre space is more specific than wavetable synthesis which can reproduce arbitrary sounds where the equivalent model would be intractable to compute. Within that space, however, a single model may capture intrinsically what would otherwise require an extrinsic definition through an unrealistically large number of wavetables. As in resynthesis with spectral modelling, the physical approach is extendible to the simulation of systems with physically unrealizable characteristics, for novel but intuitive musical effects.

2.2.3 Abstract Algorithm

Musical tones may be reproduced from recordings, or from modelling the physics of instruments or the evolution of the audio spectrum perceived by the listener. 'Abstract' algorithms do not use natural processes as a reference point in this way. Many numerical algorithms have musical properties by accident, which can be exploited for music synthesis. There chief motive is cost-effectiveness; an abstract algorithm may provide impressive sounds and yet be efficient to map into silicon: an argument that has weakened over the years with increasing VLSI performance. Frequency Modulation (FM) synthesis is the most successful example of this category (Chowning, 1973): an alternative technique is 'discrete summation formulae' (Moorer, 1976). In the simplest form of FM, a carrier sine oscillator at a frequency ω_c is frequency modulated by another oscillator at ω_m with the amount of modulation controlled by the 'index' $I(t)$ as in equation (2.1).

$$y(t) = A(t)\sin(\omega_c t + I(t)\sin\omega_m t) = A(t)\sum_n J_n(I(t))\sin(\omega_c t + n\omega_m t) \qquad (2.1)$$

The spectral decomposition of an FM waveform is a set of sinusoids shaped by a spectral envelope $A(t)J_n(I(t))$ where J_n is the Bessel function of order n. $A(t)$ governs amplitude, the index $I(t)$ governs timbre; at $I(t)=0$, the note is simply the carrier sinusoid, as $I(t)>1$ the envelope takes the form of two peaks either side of ω_c which gradually migrate causing harmonics near ω_c to diminish in amplitude and those further away to increase. Thus, the concentration of harmonic power in the spectrum varies monotonically with $I(t)$: a 'brightness' control that can be enveloped, for instance, to simulate the characteristic "waah" of brass instruments (Higgins, 1990). Another control parameter is the ratio of ω_c to ω_m: an integer ratio gives harmonicity, and a non-integer ratio, inharmonicity (for bell-like tones). The 'folding-over' of negative frequencies creates additional spectral complexity (Moore 1990).

The application of FM principles is scaleable beyond the simple oscillator pair of equation (1.4) With six oscillators per voice (as in the celebrated Yamaha DX-7) the number of possible configurations proliferates, providing a richer sound palette than AS

with the same resources. The key to the computational efficiency of FM is that modifications to the parameters of a single modulating oscillator controls *many* spectral components in a musically useable way, unlike the one-to-one relationship in AS where modifiying a single partial parameter usually has little impact on timbre (it is useful to classify respective parameter behaviour as 'strong' and 'weak') (Jaffe, 1995). However, musicians are still limited to synthetic, characteristically "FM"-type sounds and parameters that often have counter-intuitive effects compared to those manifest on acoustic instruments: a fact that complicates analysis support: genetic algorithms are proving useful in deriving FM parameters that map on to analysed acoustic timbres (Horner et al, 1992).

In contrast to FM, which uses non-linearity in oscillator phase-accumulation, 'waveshaping' creates spectra by applying a non-linear function directly to a sinusoid at the fundamental frequency. Functions for arbitrary (but strictly harmonic) spectra are derived from applying Chebyshev polynomials (De Poli, 1983). Timbre is independent of frequency, but a function of amplitude which evolves towards the specified spectrum at unity amplitude, and distorts beyond. Waveshaping principles overlap with the domain of physical modelling and, indeed, have direct application because non-linearity is a characteristic of the physics of many natural systems. For example, the vibration of a reed is linear at low amplitudes and non-linear at higher (Rodet, 1992).

2.3 Developments in Spectral Modelling

The most successful spectral modelling approach prior to the availability of commercial digital music synthesisers was 'subtractive synthesis': a complementary philosophy to AS. A harmonically rich signal, such as a pulse stream is shaped by applying a filter with an appropriate frequency response. Its origins lie in its ease of construction using analogue electronics with voltage-controlled oscillators, amplifiers and filters (VCO, VCA and VCF). 'Algorithms' were represented by hardwiring modules to create a 'patch' (Smith, 1991). It was a popular technique that became dated for two reasons, (1) simulations of acoustic instruments were judged poor and became superseded by sampling and wavetable technology which derived their sound directly from the acoustic source in question, (2) the spectral envelope of synthetic sound is shaped by low-order

filters and complex envelopes must be built up by adding more modules to a patch. FM synthesis, in contrast, can create novel and complex spectral effects with a simple oscillator topology. The sound, however, remains musically distinctive and a revival of interest in subtractive synthesis is underway as evidenced by (Stilson and Smith, 1996)

Pioneering experiments using AS on digital computers were conducted by Risset (1965) in the language Music V: it was demonstrated that a complete trumpet note could be analysed into a set of partials and satisfactorily re-synthesised using PWL approximations of partial amplitude envelopes (see section 1.1.3). Establishment of the viability of digital AS led to a concentration in providing analysis support to permit the spectral modelling of arbitrary acoustic sources. Due to the limited performance of contemporary mainframes, processing was performed off-line - often overnight - to create recordings that could be played at normal speed (Smith, 1991). Real-time digital synthesis was not yet feasible. In the early days therefore, the computational expense of AS was a matter of inconvenience, and of lesser importance compared to the power of spectral modelling to create any conceivable sound.

The simplest analysis method is the 'heterodyne filter': a bank of linearly-spaced bandpass filters implemented by frequency shifts and parallel instantiations of a prototype low-pass filtration. A synonymous technique is the digital 'phase vocoder': a sliding Discrete Fourier transform (DFT) which produces phase and magnitude values for every frequency bin each sample period. The DFT is efficiently implemented by the Fast Fourier Transform (FFT). A sinusoid appears as a magnitude peak in one of the bins, and its phase, frequency (from the sum of the inter-frame phase derivative and bin centre frequency) and amplitude parameters may be extracted. Linear bin spacing matches the linear spacing of partials in a harmonic tone. With an FFT of sufficient length, individual partials can be identified by a tracking algorithm which extracts their amplitude, frequency and phase envelopes. Over the years, many improvements have been proposed for the phase vocoder (e.g. optimisation of hop size, window function) and it remains an important analysis / synthesis tool for computer musicians (Moore, 1990).

Availability of analysis support and libraries of analysed tones provided researchers with data to investigate data-reduction methods to make AS more efficient. Early AS engine

designs such as (Snell, 1977) introduced the TOB model of Fig. 1.2 which uses PWL envelope compression. Sasaki and Smith (1980) use a similar design, but propose codifying control information as a pair of two-dimensional arrays for (i) spectral envelopes and (ii) partial frequency ratios where one subscript is a scalar 'control index' which extracts the required instantaneous control data vector for the set of S sinusoids. Additionally, there are global parameters of amplitude and frequency. Timbral evolution is effected by applying an envelope to the control indices in a similar way to that used in FM synthesis. A unique feature is the rejection of PWL in favour of low-pass filtration (f_c=100Hz) on envelope streams enabling a low-bandwidth control rate without audible artefacts. Hence, the undefined controllability of AS is resolved by making its control interface 'look' like that of contemporaneous abstract algorithms. However, all control data must be generated *a priori*. Also, generality is necessarily lost by making timbre a function of two control parameters. However, an important concept is introduced into AS for the first time: a single 'strong' parameter or 'metaparameter' is mapped onto a set of 'weak' partial amplitudes (Jaffe, 1995). This is a key strategy for resolving control issues in AS at layers of data abstraction higher than PWL representation.

Given a musical tone expressed as a large set of envelopes in PWL form, another strategy for solving the control overhead is to abstract a higher-level description. Reasons are twofold: further data-reduction is desirable to reduce implementation costs whilst derivation of intuitive 'handles' or metaparameters on the data set improves SME ergonomics. Charbonneau (1981) proposes a data-reduction technique. An amplitude envelope (normalized to 1) of all partials, and the peak value of each partial, is extracted using the product of the two for resynthesis. The same algorithm was applied to frequency functions, and each partial had independent start and finish times between which master envelopes were warped. For short duration tones, satisfactory results are reported, but the scheme does not account for temporal evolution of spectrum over longer durations. In contrast, Strawn (1980) developed a scheme, using pattern recognition techniques, for decomposing an arbitrary PWL function into a hierarchy of diminishing triangles, representing perceptually important 'features' and 'sub-features': it is the abstraction of a metastructure. Redundant information is required to express the hierarchical structure, but modifications can be made at different levels of abstraction i.e.

the overall shape may be changed, or just a single feature. Elsewhere, Strawn (1987) discusses the PWL modelling of transitions between notes in an analysis context.

Schindler (1984) uses a combination of the previous techniques to generate a hierarchy of spectra describing the relative amplitude of features and subfeatures in each partial, with relation to breakpoints on a master envelope. The advantage is that the spectral evolution of a sound is described in a structured form, based on master amplitude and frequency envelopes. This is then integrated with a two-dimensional timbre frame associated with an event e.g. attack as a function of pitch and loudness. The frame is organized into a rectilinear grid with context-specific quantisation of axes. Each point on the grid is associated with a structured envelope set and resynthetic envelopes at an intermediate point are interpolated from the vertices of the bounding rectangle. He then develops methods for timbral evolution within frames, splicing of frames for a chain of events (attack, decay), and proposes a custom architecture. The objective is to use a modifiable, structured representation - after Strawn(1980) - in a data-efficient form (after Charbonneau) with an emphasis on generality.

Kleczkowski (1989) developed Group Additive Synthesis as a data-reduction technique by identifying redundancy within sets of envelope functions. His technique depends upon clustering envelopes into groups, on the basis of Euclidean distance in raw-data form, and extracting a master line-segment amplitude and frequency for each group. Grouping methods are discussed by Eaglestone and Oates (1990) and Horner and Cheung (1995). To discriminate between partials in a group, two vectors of constants describe the relative magnitude of partial amplitude and frequency in relation to the master envelopes. He qualifies his procedure with listening tests to support his hypothesis that only perceptually insignificant data is lost during grouping. As redundancy is eliminated, a more compact control data set is extracted that is simple to uncompress in real-time. Also, hardware savings are possible if a group contains n harmonic partials: a single wavetable oscillator initialised by the IFT of the amplitude vector replaces n sine oscillators. It is also envisaged that the use of master envelopes makes grouping compatible with the metastructures employed by Strawn (1980) and Schindler (1984).

Another strategy is to prune out weak partials that are psychoacoustically 'masked' by stronger ones in their frequency vicinity. The fundamental idea can be understood in an AS context via a simple conceptual experiment: a listener is presented with two simultaneous tones; (i) a reference sinusoid of constant amplitude and frequency at f_1 and (ii) a sinusoid of variable amplitude and frequency at f_2. The graph of amplitude(f_2) versus f_2 defining the threshold where the listener perceives both tones, rather than just the reference f_1 (i.e. f_1 is just masking f_2), is the 'critical band' at f_1 which takes the characteristic form of a bell-shaped curve illustrating the bandpass excitation pattern caused by f_1 along the basilar membrane in the cochlea. Below this threshold, f_2 is not perceived as a separate tone and its presence, or otherwise, is perceptually irrelevant. In reality, the interaction (beating) of f_1 and f_2 would be noticed because of the finite time resolution of the ear thus reducing, somewhat, the universal validity of such a simplified method for high-quality AS. Returning to a high-level perspective, the idea of 'receiver coding' is that, given (a) foreknowledge of the critical band properties of the inner ear (via an auditory model) and (b) a partial set described by AS amplitude and frequency vectors, a reliable prediction can be made as to which partials will be masked and thus not require synthesis (Moore, 1990).

An example of the successful exploitation of masking is the MPEG algorithm for high-fidelity audio data-compression at low bit-rates. The bit-rate (and thus quantisation noise) in each sub-band is optimised to shape the full-band noise-floor to the expected masking envelope of the full-band signal spectrum so that the perceived signal to noise ratio remains acceptable (Pan, 1994). However, receiver coding of time-varying PWL AS parameters is a different issue. Savings made by minimising the number of active oscillators must be in excess of the pruning overheads for the technique to be economic. An evolving spectrum implies (i) frequent (and expensive) recomputation of the auditory model and (ii) the high probability of intermittent partial masking necessitating frequent oscillator allocation and de-allocation (increasing resource allocation complexity for little re-allocable gain). A simplification is to assume that masking is constant during the duration of a note because of inherent frequency stationarities (c.f. section 1.4.1).

For instance, Marks (1988) allocates static priorities related to partial amplitudes; the higher the value the more significant its contribution is to the perception of timbre. These are used to prioritize resource allocation in a low-complexity AS engine with a value of S insufficient to handle the theoretical peak number of partials. Haken (1991) develops this idea further by introducing a simplified auditory model to predict masking effects. The spectrum is decomposed into a series of seven frequency bins approximating critical band spacing. Partials are mapped into bins and are pruned to a fixed number, if such is exceeded, on the basis of retaining those with maximum peak amplitude. In listening tests, it is argued that by exploiting masking effects no more than 75 partials are required for the synthesis of a stationary "symphonic" chord. A real-time implementation uses timbral interpolation within a latticed cubic space (see section 2.2.1). Of interest to this thesis, two sample rates are utilised, with low frequency partials synthesised at the lower sampling rate showing a latent recognition amongst researchers in the application of multirate DSP to AS.

Provided with a group of instruments sounds, it is possible to identify a set of basis functions for additive modelling of the group of smaller size than the equivalent set of sinusoids. Stapleton and Bass (1988) demonstrated that the Karhunen-Loeve (KL) transform can generate such a set to span a group of instrument waveforms that share a strong cross-correlation. Linear combinations of the basis functions reconstruct the individual timbres of the group. Plomp (1976) used similar principles in extracting an optimal set of spectral shapes, in frequency domain. Listening tests confirm the quality of resynthetic tones. The chief advantage reported is the replacement of an unspecified number of sinusoids by a maximum of five basis functions, reducing the amount of real-time computation and control data. Disadvantages include (1) a limitation to harmonic tones and (2), the spectrum defined by a basis function can alias at high frequencies and must be bandlimited to allow frequency variation.

The 'Sine Circuitu' (Jansen, 1991) is a typical recent example of a TOB form; a single module can generate 625 sinusoids at f_s=44.1kHz, with linear amplitude and frequency envelope generation under the control of a transputer. Up to sixteen modules can be connected in parallel to provide 10,000 sine generators in real-time. An object-oriented

environment for algorithm development is provided. Such machines are invaluable test-beds for researching the potentials of real-time AS. The chief disadvantage is cost. A number of high speed discrete logic components are required, and the control mechanism is therefore simplified to an operational minimum; discrete integration for enveloping without hardware breakpoint detection or line segment queuing. These low-level tasks are transferred to the controlling transputer software. In contrast, a recent example of a small-scale prototype TOB is described by (Houghton et al, 1995). It is based upon a (standard cell) ASIC, containing control logic and an interpolated LUT (its chief feature of academic interest, see section 7.4.3) with an external multiplier / accumulator and memory, mounted on a plug-in board to an IBM PC which provides real-time control. 127 oscillators are thus supported with envelope generation performed in software.

An early example of the commercial exploitation of digital AS is the BMIS (Bradford Musical Instrument Simulator) developed at the University of Bradford, UK (Comerford, 1993). It is a general purpose architecture comprising up to eight 'Music Modules', each offering 64 'hardware generators'. Simulation of classical pipe organs is the chief application. The internal architecture of a module differs slightly from the standard pattern by storing waveforms in RAM instead of a sine LUT in ROM. A 'primary waveform synthesizer' writes the required waveforms into RAM at run-time from spectral data sent to the module from a host computer via the 'Musibus'. Savings in computation are achieved in a variety of ways as discussed by Marks (1988) and Kleckowski (1989). Also, a set of partials coincident in frequency from consonant voices in a chord can be replaced by a single sinusoid of equal RMS power to the set.

Spectral Modelling Synthesis (SMS) as developed by Serra and Smith (1990) is currently one of the most popular tools for the music analysis / synthesis. AS models the deterministic part of the signal (composed of a sum of sinusoidal partials) whereas subtractively filtered noise (via an IFFT) models the stochastic part, once the former has been isolated by a partial tracking algorithm. Accurate modelling of sounds which have a strong noise component (e.g. human speech) is thus facilitated. As originally proposed, SMS assumed an oscillator bank model of AS, but its compatibility with AS via the IFFT using the method of Rodet and Depalle (1992) leads to a logical integration of the

stochastic and deterministic components into a single IFFT-AS schema. This proprietary technology is known as Fourier Analysis Resynthesis (FAR) (White, 1995).

2.4 IFFT Simulation of the TOB

The radix-2 IFFT is a widely-used optimised form of the Inverse Discrete Fourier Transform (IDFT) which converts a discrete frequency vector $X(\omega)$ of N bins (where N is an integer power of 2), each describing the magnitude and phase of bin frequency in complex form, into a signal vector $x[m]$ of N samples. For real-time synthesis of a continuous music signal, consecutive IFFT's or 'frames', are required with the Short-Time Fourier Transform (STFT) of the desired signal written into an IFFT each frame where N is chosen to give good time resolution. Use of the term STFT is justified by the observation that N is the length of the FFT / IFFT analysis window (which is by default rectangular). The significance for AS is that a controllable sinusoid may be synthesised by predicting its STFT each frame after which N samples are automatically generated by the IFFT, in contrast to the computation required *each* sample period in a TOB. By superposition, a single IFFT can accommodate any number of sinusoids with the limitation that they are accumulated into a single sample stream. For a suitable reference of IFFT musical applications see Chamberlin (1980).

The control data rate is reduced from the sample rate to the frame rate: Rodet and Depalle (1992) propose a typical value of 128:1. Step changes in parameter values between frames generate high frequency interference that is distracting for the listener. To counteract this, smooth amplitude and frequency interpolation is required between frames to make the coarse resolution of control imperceptible. Also, control resolution has a lower bound due to Heisenberg's principle (as discussed in section 1.4.2). For instance, the synthesis of fast rise-time transients is difficult if N is too large. Hence the frame size should be a compromise between small N for sufficient control resolution and large N so that the condition in eqn. (2.2) is satisfied where c_{TOB}, c_{IFFT} and c_{STFT} are, respectively, the costs of a TOB oscillator update, IFFT computation and STFT computation for a single sinusoid (Freed et al, 1993).

$$\frac{Sc_{STFT} + c_{IFFT}}{N} < Sc_{TOB} \qquad (2.2)$$

2.4.1 Overlap-Add Synthesis

Early proposals for IFFT-AS synthesis mapped one sinusoid to one IFFT bin providing only $N/2$ harmonics of the frame frequency for synthesis (Chamberlin, 1980). Exact frequency is maintained by rounding the desired frequency to the nearest bin and introducing a phase lead or lag each frame to approximate the residual frequency fraction. With non-overlapping rectangular windows, phase discontinuities appear at the frame interfaces. However, Overlap-Add (OLA) synthesis with raised cosine windows smoothes out the discontinuities and performs the desired interpolation between the different amplitude values that a sinusoid may take (according to $A_i[n]$) in each frame. In OLA the sum of overlapping windows is always unity as illustrated in Fig. 2.1 with triangular windows. However, twice the number of IFFT's are required. The resulting signal suffers from amplitude modulation because of smeared artefacts of phase discontinuities (Chamberlin, 1980). Smearing effects are avoided in the 'FFT^{-1}' algorithm of Rodet and Depalle (1992) to which the rest of this section is devoted: for other pertinent OLA analysis-synthesis proposals see George and Smith (1992), McAulay and Quatieri (1986, 1987, 1988) and Fitz et al (1992).
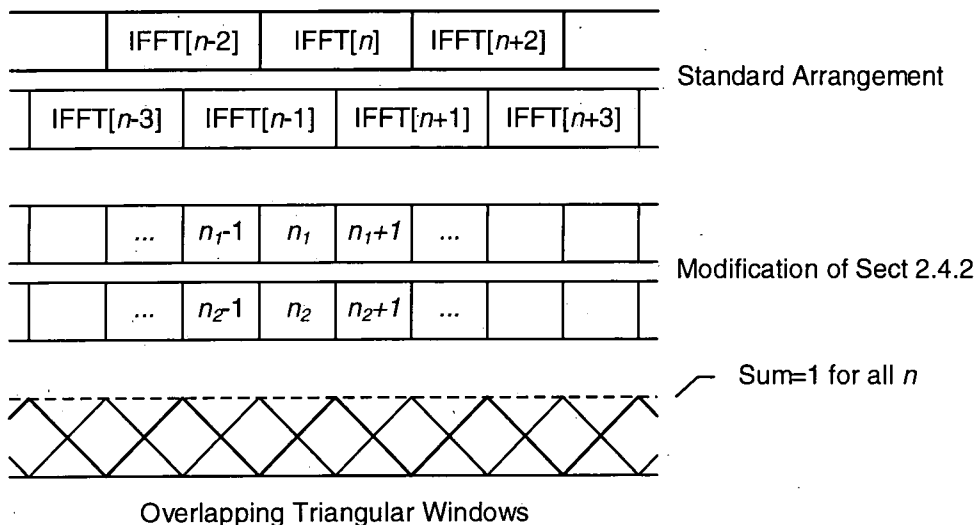
Figure 2.1   Overlap-Add IFFT Synthesis with Triangular Windows

According to the modulation theorem, a stationary sinusoid at frequency F, multiplied by a window $w[m]$, is the frequency domain convolution of their Fourier transforms; the delta function $\delta[\omega - F]$ (for a cosine) and $W(\omega)$. In other words, a displacement of $W(\omega)$ to F. By suitable choice of $W(\omega)$, most of the energy is concentrated near the maximum which is significant as only a small number of look-ups in an LUT approximation of $W(\omega)$ within a narrow margin of the main lobe are necessary to approximate the STFT of a sinusoid for arbitrary F: Brown and Puckette (1992) use a similar strategy in an analysis context. According to Rodet and Depalle (1992), FFT^{-1} uses an oversampled $W(\omega)$ table of length 512 requiring 9 STFT look-ups, and an IFFT of length $N=256$. The STFT of the resultant unity-gain cosine at frequency F is scaled to the desired amplitude A and rotated about the frequency axis so that its phase matches the previous frame which is achieved by accumulating the phase difference between frames, determined as $N\pi F/f_s$ for time-invariant F. Finally, the STFT is accumulated with those of other sinusoids prior to IFFT / OLA synthesis.

A triangular window $t[m]$ for OLA synthesis, illustrated in Fig. 2.1, performs linear amplitude interpolation between frames and is more desirable than a raised cosine as suggested by Chamberlin (1980) which, for instance, connects frame-rate quantised breakpoints by scaled versions of the curve $\sin(x)\!:\!-\pi/2 \leq x \leq \pi/2$, giving a ripple effect for a sinusoid increasing linearly in amplitude across frames. However, the Fourier transform of $t[m]$, $T(\omega)$, has significant smearing of frequency and therefore poor properties for $W(\omega)$ and so an alternative window (which has good spectral resolution) is chosen to construct the STFT of each sinusoid (e.g. Hamming): Nuttal (1981) provides a tutorial on suitable analysis windows. Finally, to ensure that the IFFT of the STFT - windowed by $w[m]$ due to its construction by $W(\omega)$ - generates a signal windowed by $t[m]$ for OLA, the output of the IFFT is post-multiplied by $t[m]/w[m]$.

2.4.2 Supporting Frequency Envelopes

The inclusion of frequency enveloping in the FFT^{-1} algorithm requires a mechanism that performs frequency interpolation between consecutive frames (from $F_x[n-1]$ to $F_x[n]$) for a sinusoid x. A solution proposed by Goodwin and Rodet (1994) is to synthesise chirps within frames; nonstationary sinusoids changing linearly in frequency from $F_x[n-1]$ to

$F_x[n+1]$: such end-points are determined by the OLA algorithm. The STFT of a chirp is no longer a simple displacement of $W(\omega)$, but a broader version with a breadth dependent on $\Delta_F = F_x[n+1] - F_x[n-1]$ and centred upon $(F_x[n+1] + F_x[n-1])/2$. As the distortion of $W(\omega)$ is dependent only on Δ_F, and $W(\omega)$ is a compact LUT, a two-dimensional array $W(\omega, \Delta_F)$ is a convenient representation: a coarse resolution (in terms of Δ_F) with linear interpolation is favoured. Once $W(\omega, \Delta_F)$ is derived and displaced to $(F_x[n+1] + F_x[n-1])/2$, the processing steps are identical to the case of a stationary sinusoid. Maintaining correct inter-frame phase, however, requires a quadratic formula to take into account the parabolic phase trajectory of the chirp.

The proposed modification assumes that Δ_F is constant in consecutive frames. If Δ_F changes between frames, then a splicing error occurs because of an irreconcilable phase mismatch between the different chirp rates in the overlap region: discontinuity side-effects of IFFT / OLA synthesis re-emerge in the first-derivative of $F_x[n]$. A further refinement proposed by Goodwin and Kogon (1995) that transposes the error to the second-derivative is to divide frames in two and match the chirp rate in the first half of a successor frame with that in the second half of its predecessor i.e. $\Delta_F = F_x[n] - F_x[n-1]$. For linear amplitude interpolation between frames, a chirp of amplitude $A_x[n-1]$, multiplied by the falling half of $t[n]$, is summed with a chirp of amplitude $A_x[n]$, multiplied by the rising half of $t[n]$. Two IFFT's of length $N/2$ are thus required to permit the superposition of independently controllable sinusoids as illustrated in Fig. 2.1. The same STFT is used for both chirps of x, scaled to $A_x[n-1]$ and $A_x[n]$ for the IFFT pair.

2.4.3 Noise Synthesis

Noise synthesis is straightforward with IFFT / OLA synthesis. As in SMS (Serra and Smith, 1990), the magnitude spectrum of the desired noise is written into an IFFT with random phase each frame to avoid periodicity. SMS models the residual noise envelope as a PWL spectral envelope each frame with equispaced breakpoints in frequency. This is permissible as the noise envelope is smooth and peaks, representing sinusoids, are extracted beforehand by a partial tracking algorithm. Residual noise representation can be minimised further by quantising the spectrum into Equivalent Rectangular Bandwidths (ERB's) via an auditory model and computing the noise power required in each ERB

Page 34

(Goodwin, 1996). By superposition, both noise and sinusoidal synthesis may be integrated in the same IFFT.

2.4.4 Performance Evaluation

FFT^{-1} is primarily a software algorithm due to the complexity of operations required to generate the STFT of nonstationary sinusoid. For maximum performance, attention is paid to the instruction and data tables sizes to ensure that critical sections of the algorithm are accommodated in the CPU caches and accesses to slow external memory are minimised. According to Freed et al (1993) implementation of FFT^{-1} in 'C' on a MIPS R4000 workstation (internal clock 100Mhz) shows that a maximum of 310 oscillators are supported at f_s=44.1kHz whereas a software oscillator bank yields 55: a speedup of approximately 6. Freed et al (1993) reasons about the inefficiency of VLSI solutions and discusses a hypothetical 50Mhz monolithic VLSI oscillator bank requiring 4 clock cycles per update. This is calculated to yield a maximum of 290 sinusoids. The conclusion is drawn that cost-effective AS resources are most efficiently provided by a combination of efficient coding of FFT^{-1} (via optimising compilers) and high-clock rate CPU's with sufficient caching to support the code and data.

The STFT operations in FFT^{-1} represent a pre-processing kernel which takes the standard PWL AS envelopes of $F_i[n]$ and $A_i[n]$ and transforms them into a form suitable for IFFT / OLA synthesis. The interface is cast to resemble a TOB except that breakpoints are quantised to the frame rate of $2f_s/N$ rather than f_s. The optimality of FFT^{-1} has an upper bound because increasing N implies a finer resolution IFFT which, in return, requires a finer resolution of $W(\omega)$ in the STFT to maintain synthesis quality. However, T_{max} from section 1.2.2. places an upper bound on N because of delays, proportional to N, in blockwise IFFT processing (Jaffe, 1995). However, the requirement for sufficient control resolution as discussed in section 2.4 - which is finer than that of T_{max} - means that latency is not a significant problem with FFT^{-1}.

One method for representing control data is to superimpose PWL enveloping over a coarser time grid. For line-segments longer than a frame, control data for intermediate frames is interpolated at frame-level. However, PWL presumes that the next breakpoint is known *a priori*, precluding gestural control by metaparameters (Jaffe, 1995). In

contrast, the technique suggested for FFT^{-1} is to store a file of spectral envelopes indexed by a metaparameter (Sasaki and Smith, 1980) for compatibility with analysis tools such as SMS which convert a musical signal into a sequential file of spectral frames describing the noise spectrum and a set of oscillator frequency and amplitudes. If the analysis file is indexed sequentially, then exact resynthesis occurs. To effect certain key time-domain transformations, however, linear interpolation between frames *is* required. One example is time-stretching without altering pitch. Hence, IFFT-AS has the net effect of creating a coarser PWL time grid than that presumed by the TOB. A single low control rate of f_s/N is imposed in place of a higher, redundant rate of f_s.

2.5 Review

In outlining the literature relating to AS, it emerges that though the definition of AS is succinct - eqn. (1.1) - the question of control remains open-ended. Researchers have addressed this control problem from two different angles; (i) spectral modelling paradigms for supporting analysis parameters in a data-efficient, modifiable and intuitive manner and (ii) optimising oscillator bank design, with an intermediate representation between these layers of PWL envelopes. This situation displays little evidence of change, and major future innovations lie in the area of making implementations more powerful, useable and affordable. A variety of strategies have been proposed, some of which compromise generality - a fundamental advantage of AS - for apparent performance gain. From the evidence presented, the conclusion is drawn that the long-term application of an AS data-reduction technique is related to how generally it can be applied.

3. Multirate Additive Synthesis

3.1 Overview

As concluded in section 1.4.3, an analysis of AS computation in the context of note-based music leads to the notion of computational optimisation by Multirate Additive Synthesis (MAS). To recap, the two chief areas of computational burden in MAS are

1. Interpolating S decimated sinusoids with independent sample rates up to f_s
2. The real-time scheduling of S sample rates (in all probability incommensurate).

The aim of this chapter is to develop the theoretical basis for a MAS algorithm by applying classical multirate concepts to the problems posed by (1) and (2). The conclusions of Chapter 1 are accepted *per se* (especially sections 1.2, 1.3.2 and 1.4.3) and form a starting point or 'requirements specification' to the task in hand. It transpires that an efficient solution for (1) leads to a great simplification in (2), which is finally solved in the proposed MAS Coprocessor (MASC) design of Chapter 8: discussion in this chapter is thus confined chiefly to topic (1).

Chapter 2 concluded that FFT^{-1} (Rodet and Depalle, 1992) is the contemporary 'state-of-the-art' in AS implementation because it did not compromise the goals of section 1.2 and is software-oriented. The well-spring for this chapter comes from the common knowledge that FFT / IFFT and multirate approaches are complementary methods for DSP optimisation and that, as yet, the latter has not been exploited systematically for AS optimisation, though there is latent interest (Haken, 1991). However, musical applications of multirate DSP outside the sphere of AS are discussed by Holm (1994) and Levine (1996a, 1996b). The fundamental premise for this chapter is that, in contrast to frequency-domain FFT / IFFT approaches, multirate techniques are time-domain and hence more compatible with a TOB architectural model; the optimal computational model for eqn. (1.1) as discussed in section 1.3.2.

3.1.1 Interpolation and Decimation in Multirate DSP
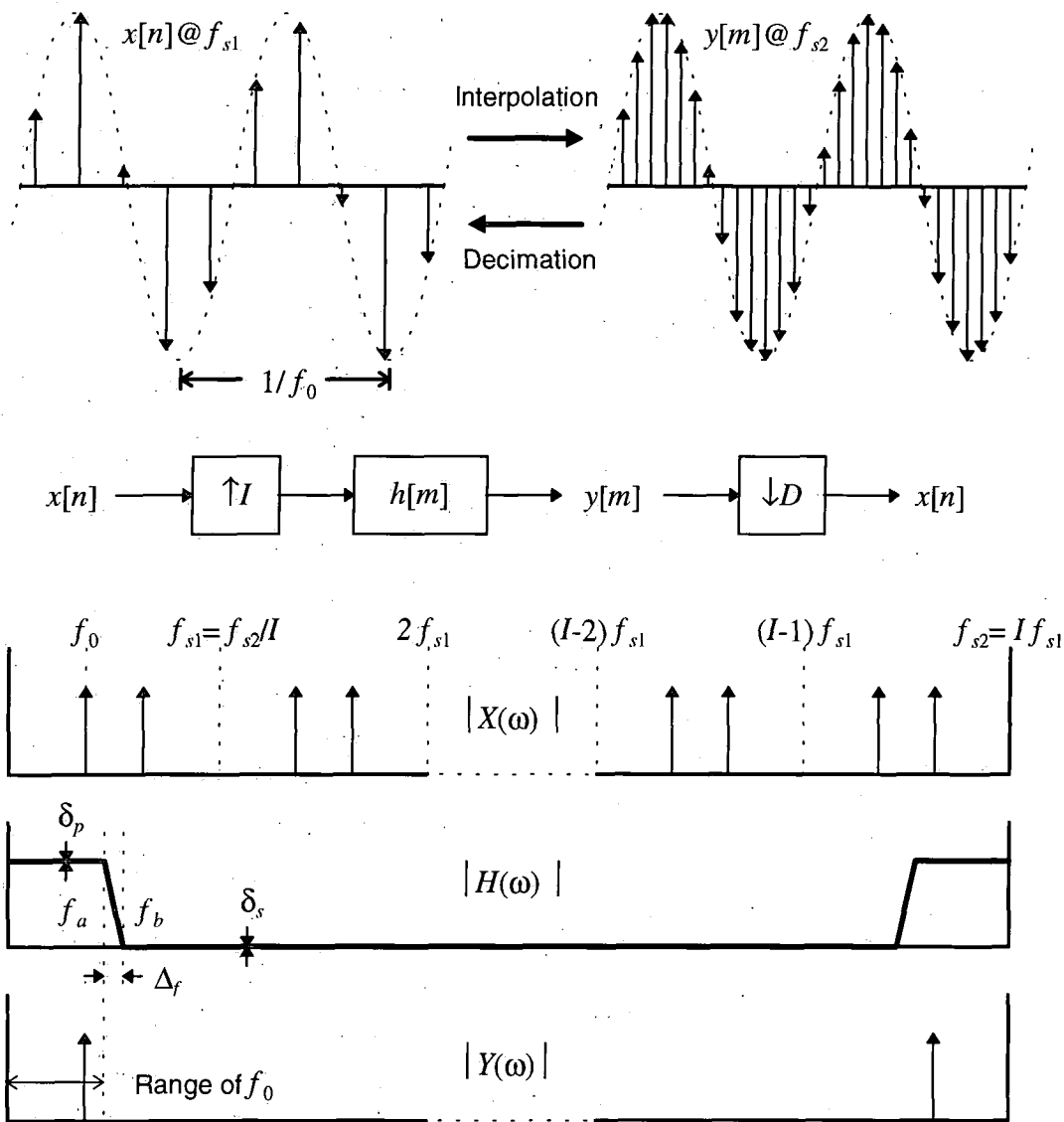
Figure 3.1   Interpolation and Decimation of Sinusoids

A definition of multirate DSP is the simultaneous processing of signals having sample rates (likely to be different) which are optimised to the signal bandwidths in question (Crochiere and Rabiner, 1983), (Vaidyanathan, 1993). A method enabling sample rate changes, without affecting signal content, is required. Two basic operations are therefore fundamental to multirate DSP: *interpolation* is an increase in sample rate by a factor I and *decimation* is a decrease by a factor D. Fig. 3.1 illustrates how these operations relate to a sinusoid in both time and frequency domain, since this is relevant to AS. A

stationary sinusoid $x[n]$ at frequency f_0 and sample rate of f_{s1} is interpolated by I (integer valued) to $y[m]$ at a new rate of $f_{s2}=If_{s1}$, which is then decimated by $D=I$ to recover $x[n]$ showing that decimation is the inverse of interpolation.

Conceptually, the change from f_{s1} to f_{s2} is achieved by inserting I-1 zero-valued samples between each sample of $x[n]$. This has no effect on the signal spectrum in that images of the signal baseband (i.e. between DC and $f_{s1}/2$) occur about harmonics of f_{s1}. However, the frequency response images of a filter ($h[m]$) executed at f_{s2} occur about harmonics of f_{s2}. $H(\omega)$ is a low-pass filter which passes only the baseband component of $X(\omega)$ (self-evidently with those about harmonics of f_{s2}) and suppresses all intermediate components about harmonics of f_{s1} in order to generate $Y(\omega)$. These components constitute noise due to time quantisation. If f_{s2}>40kHz, then audible components up to $f_{s2}/2$ are removed leaving a pure audio frequency sinusoid in the baseband. Relevant to MAS is the observation that, in an AS oscillator bank, the decimated sinusoid $x[n]$ has a computational ratio of 1:I in comparison to its oversampled equivalent $y[m]$.

For $H(\omega)$ to have an economic implementation, a non-ideal design must be used which is parameterised by three tolerance variables determining performance; δ_s and δ_p set upper bounds on, respectively, passband and stopband ripple (gain relative to the ideal passband gain at 0dB) and f_a and f_b mark a finite transition region from stopband to passband where the frequency response gradually rolls off, often quoted as $\Delta_f=(f_b-f_a)/f_{s2}$. The relationship $f_a=f_{s1}-f_b$ places Δ_f symmetrically over the Nyquist frequency at f_{s1} of $f_{s1}/2$ such the stopband of $H(\omega)$ exactly coincides with range of f_0 harmonics requiring suppression. For high-quality interpolation with a flat frequency response, the range of f_0 is restricted to $0 \leq f_0 \leq f_a$ i.e. the passband of $H(\omega)$. It is undesirable for f_0 to enter Δ_f because (i) it is attenuated and (ii) a mirror-image harmonic in $f_{s1}/2$ is simultaneously boosted, with a risk of becoming perceptible. The range of f_0 in $x[n]$ is therefore oversampled by $f_{s1}/2f_b$ compared to an ideal H(ω).

I must be an integer for $h[m]$ to be a time-invariant filter. Conversion of non-integer ratios of f_{s2}/f_{s1} requires time-varying filtration with a resampling of the coefficients in $h[m]$ from a continuous time impulse response each sample period. If f_{s2}/f_{s1} is rational, in that they are synchronous and originate from the same clock, then a finite cycle of

coefficients of length $f_{s1} f_{s2}/c^2$ suffices, where c is the highest common factor of f_{s1} and f_{s2}. This can create a large cycle of coefficients with high storage demands making resampling from a high resolution look-up table more economic. An alternative is interpolation by I followed by decimation D such that $I/D = f_{s2}/f_s$: interpolation can use an efficient polyphase structure, outlined in the next section, with decimation implemented by commutating, modulo-D fashion, around the sub-filter outputs. An irrational ratio of f_{s2}/f_{s1} requires a complete recomputation of coefficients each sample period. This situation occurs when f_{s1} and f_{s2} are asynchronous and sourced from different clocks when linking equipment with incommensurate sample rates (e.g. CD at 44.1kHz and DAT at 48kHz), requiring a specialised convertor (Adams and Kwan, 1994).

3.1.2 Efficient Interpolator Design

There are two efficient alternatives for implementing interpolators with integer values of I that are pertinent to outline for future discussion, though the reader is referred to the literature for an in-depth analysis (Vaidyanathan, 1993). The first is the *multistage interpolator* which splits the interpolation process into its prime factors $I_1, I_2.. I_N$ such that $I = I_1 \times I_2.. \times I_N$. Each stage is unique, with passband and stopbands adjusted to correspond precisely to sidebands in the signals that require elimination. Moving up the cascade, relative transition widths increase as the signal becomes increasingly oversampled. Transition regions are of maximum width, and computation is minimised in that high-cost sharp roll-off filters are executed at a lower sampling frequency than low-cost gradual roll-off filters. The disadvantage of multistage interpolation is that it is limited to values of I that decompose into a convenient set of prime factors.

An alternative to the multistage interpolator is a single-stage 'polyphase' interpolator which functions for any integer value of I. The output $y[m]$ is taken by commutating between I 'subfilters' h_i $(0 \le i \le I-1)$ each sample period of $x[n]$ where each h_i is, conceptually, an allpass filter introducing a pure phase delay of $2\pi i/I$. Only the subfilter connected to the commutator 'rotor' is computed each sample period of $y[m]$. The cost of computation reaches an asymptotic limit with increasing I because the complexity increase is accommodated by a proliferation of sub-filters with the order of each $H_i(z^I)$ remaining constant for a given (δ_p, δ_s). If the prototype interpolation filter $H(z)$ is FIR,

$H_i(z^I)$ $(0{\leq}i{\leq}I\text{-}1)$ corresponds to the modulo-I activation of coefficients caused by samples of $x[n]$ shifted into $H(z)$, separated by I-1 zero-valued samples

3.1.3 A General Model for MAS using Subband Decomposition

Consider the interpolation scheme of Fig. 3.1 modified such that we may exploit the computational savings implied by eqn. (1.4) by synthesising x in a baseband spanning DC to f_a where $f_{s1}{\cong}f_{opt}(x)$ (assuming an ideal H(ω)). x is then interpolated by $H(\omega)$ and heterodyned by $f_{min}(x)$ to the desired spectral location. For S independent oscillators, a unique instantiation of this scheme is required for each oscillator if the optimal value of $f_{opt}(x)$ were to be applied strictly leading to a costly and complex implementation. The reduction in oscillator update rate by using $f_{opt}(x)$ in place of f_s is outweighed by the cost of a high-quality interpolator executing at a rate of f_s. Furthermore, to restate the points of section 3.1 in the light of section 3.1.1, each optimum interpolation ratio $f_s/f_{opt}(x)$ is likely to be irrational leading to (i) structural complexity of individual interpolators and (ii) the problem of scheduling S incommensurate sample rates within a hard real-time context of T_{max}. S individual interpolator designs are also required. In contrast, consider the simplicity of the 'round-robin' schedule of the TOB in section 1.3.2. However, analogue time-varying (or 'tracking') interpolators for each voice are used successfully in samplers because the number of voices is low in comparison to AS (see section 2.2.1).

The solution is to use a finite set of K interpolators / frequency shifters which constitute a set of 'subbands' in frequency domain denoted $s_k{:}0{\leq}k{\leq}K\text{-}1$, each defined by a unique set of bounds $\{f_{min}(s_k), f_{max}(s_k)\}$: such a set of interpolators summed into a single output at f_s is called a *synthesis filterbank.*. Each s_k is characterised conceptually by outputting $f_{min}(s_k)$ when an input oscillator is at DC, and $f_{max}(s_k)$ when the input is at the Nyquist rate of $f_s(s_k)/2$ and therefore includes the required frequency shift. Therefore the interpolation factor $I(s_k)=f_s/(f_{max}(s_k)\text{-}f_{min}(s_k))$ and $f_s(s_k)=2(f_{max}(s_k)\text{-}f_{min}(s_k))$, assuming ideal filters for convenience. The best subband for a partial x is that which has the lowest value of $f_s(s_k)$ and yet satisfies $f_{max}(s_k){>}f_{max}(x)$ and $f_{min}(s_k){<}f_{min}(x)$ ensuring that x will not alias during note lifecycle, if the *a priori* values for x are correct. However, the computation of $\Phi_i[n]$ and $F_i[n]$ assumed for eqn. (1.1) at f_s require linear scaling to $f_s(s_k)$ - the subband context - such that x appears at the output at f_s with correct phase and frequency. The $I(s_k)$'s may

now be preset to convenient integer-only values since they are independent of $\{f_{min}(x),$ $f_{max}(x)\}$ leading to efficient filterbank implementations (as in section 3.1.2) and a set of easily scheduled oscillator sample rates.
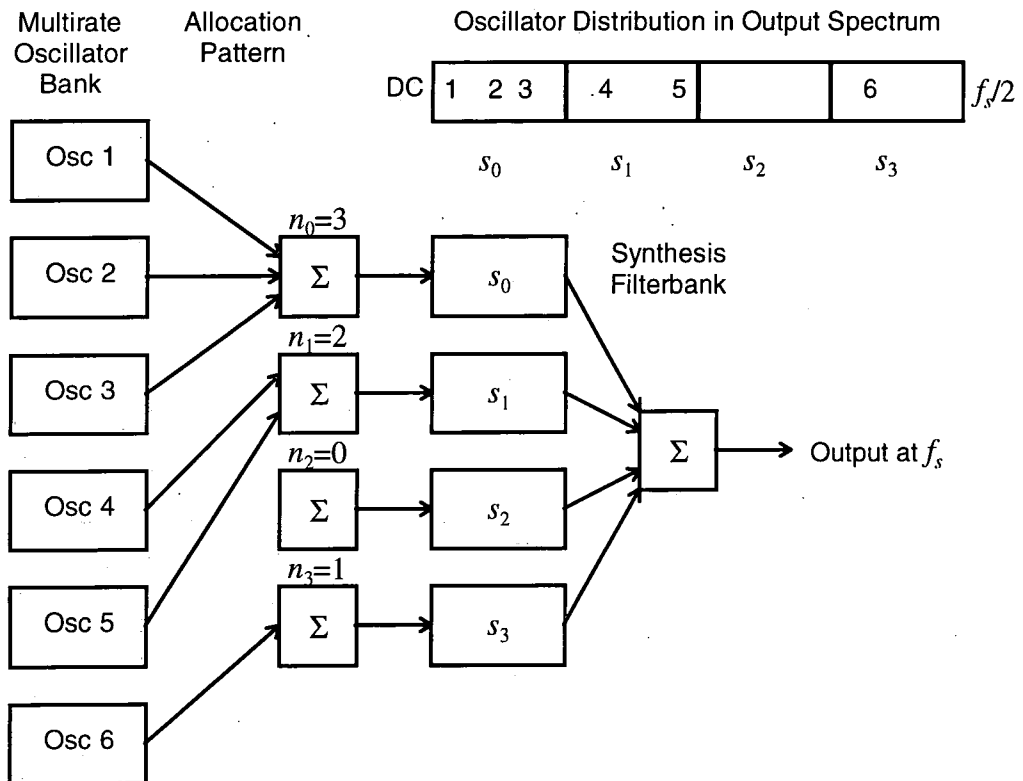
Figure 3.2   A General Model of Multirate Additive Synthesis

For illustration, a simplified example of MAS utilising a subband decomposition is shown in Fig. 3.2 with a scheme of four integer-spaced subbands (and hence four interpolators, each of $I=\uparrow 4$). The single summation of eqn. (1.1) is factorised in two by the introduction of K intermediate interpolators, utilising the superposition property of interpolation in that N parallel instantiations of a single prototype interpolator with post-summation is functionally equivalent to pre-summation followed by a single interpolator. Associated with each s_k is a set of n_k oscillators which execute and are accumulated at the input sample rate of subband s_k - f_s (s_k) - which interpolates up to f_s for final accumulation. When n_k is large, the interpolation overhead per constituent oscillator is small: a quantitative cost / benefit analysis is developed in section 3.1.7. Redundancy

exists in the suboptimal fit of x to s_k which can be minimised by a judicious choice of subband decomposition..

3.1.4 On Oscillator Q

$$Q(x) = \frac{f_{max}(x) + f_{min}(x)}{2(f_{max}(x) - f_{min}(x))} \tag{3.1}$$

For a partial x, a useful relationship between $f_{min}(x)$ and $f_{max}(x)$ is 'Q'; traditionally the 'quality' factor for bandpass filters (ratio of centre frequency to bandwidth). Broad-band signals are low-Q and narrow-band signals are high-Q. $Q(x)$ is calculated from eqn. (3.1) and is significant when sinusoids in the AS set are non-stationary, tending to force Heisenberg's inequality of eqn. (1.5) towards an equality. If AS tones are relatively stationary (e.g. resynthesis of a fixed-pitch resonant system with constant excitation like an organ pipe) then $Q(x)$ is not of interest. There is a duality in how $Q(x)$ relates to AS which requires some explanation. For instance, a high frequency attack transient may have a fixed centre frequency, but generate short-term side-band energy characteristic of a low-Q signal requiring a wide dilation of $\{f_{min}(x), f_{max}(x)\}$. For differing reasons, the same result is true of a meandering $F_i[n]$ envelope as $\{f_{min}(x), f_{max}(x)\}$ is normalised to envelope extremities, though short-term side-band energy is negligible and the sinusoid therefore has a high-Q characteristic. This duality originates in the necessity for a time-invariant $f_{opt}(x)$ via eqn. (1.4) but both scenarios tend to form an equality out of eqn. (1.5) in that $Q(x) = c$ where c is some frequency-independent constant ($c > 2$): known popularly as the constant-Q principle (Brown and Puckette, 1992). Two examples provide an illustration:

1. For resonant systems excited by an initial transient, the partial decay rate and bandwidth is proportional (quantified by c) to centre frequency (c.f. the Karplus-Strong plucked string algorithm in section 2.2.2).

2. For the harmonically-spaced partials of a relatively stationary note undergoing pitch modulation, c is related to the modulation range.

3.1.5 Non-Overlapping Subband Decompositions

An initial scheme to minimise redundancy in the suboptimal fit of x to s_k is a fine-resolution integer series of subbands, illustrated in Fig. 3.3 for $K=16$. A key advantage is that a single oscillator sample rate suffices, removing the necessity to schedule multiple rates; analogous to the frame rate in IFFT synthesis. However, the constant-Q principle is circumscribed leading to two problems. Depending on K, there will be a threshold, denoted f_{eq}, where subband width equals the expected value of $Q(x)$: dependent on the type of note synthesised. If $f(x)$ denotes the 'operating point' of x then if $f(x) > f_{eq}$ subbands become too narrow to accommodate $Q(x)$ risking aliasing of x. For $f(x) < f_{eq}$, subbands have excessive width for $Q(x)$, leaving x oversampled implying computational redundancy. Interpolation resources are poorly distributed: the ear has a logarithmic perception of pitch and $K/2$ subbands - that is half the filterbank computation - cover the top octave of perception. In its defence, an integer series remains an intuitive mapping for the harmonically-spaced partials of a musical note, particularly if the higher partials are relatively stationary and narrow-band i.e. f_{eq} is towards the top end of the audio spectrum because $Q(x)$ is very high.
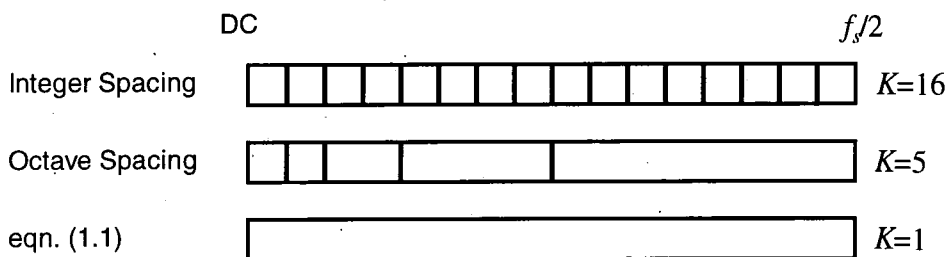
Figure 3.3   Classic Subband Decompositions

Octave subband spacing satisfies the constant-Q property and is illustrated in Fig. 3.3. An infinite series is required for true octave spacing starting from DC, and so the first two subbands are of equal width to pad the series. Two advantages of octave spacing, in comparison to integer spacing, are illustrated by the fact that it (i) uses but one interpolator (in place of $K/2$) to cover the top octave of perception and (ii) performs this at a high sample rate of $f_s/2$, sufficient for the control resolution and bandwidth required by low-Q high-frequency transients. With decreasing frequency, this difference lessens until it can be seen that the first two subbands in an octave-spaced series are integer-

spaced. To its disadvantage, octave spacing possesses progressively wider subbands implying increasing sample rates: high-Q high-frequency sinusoids are oversampled and are therefore computed more inefficiently than in an integer-spaced scheme. There is a proliferation to K-1 oscillator sample rates in place of the single rate for integer spacing indicating a potentially significant increase in oscillator scheduling overheads.

3.1.6 Overlapping Subbands

The common property of integer and octave spacing is that they are non-overlapping. The reason that they have been presented first is that they are the classical ways in which multirate DSP is used to analyse - and resynthesise - a fullband spectrum via subbands (Crochiere and Rabiner, 1983). The problem for MAS is that an oscillator x - paramaterised by $\{f_{min}(x), f_{max}(x)\}$ - cannot be allocated across a boundary between adjacent subbands. This situation occurs when, for all subbands, $f_{max}(s_k) > f_{max}(x)$ and $f_{min}(s_k) < f_{min}(x)$ cannot both be satisfied meaning that x cannot be synthesised without aliasing out of a subband. However, any x that fails these conditions can be synthesised by including eqn. (1.1) in MAS which supports arbitrary $\{f_{min}(x), f_{max}(x)\}$ if overlaid with the chosen non-overlapping subband series. However, fullband synthesis (i.e. classical AS), illustrated in Fig. 3.3 for comparison, is the most expensive of all.

An improvement is the fully-overlapping octave-spaced series illustrated in Fig. 3.4. The benefits of the octave-spaced series discussed in section 3.1.5 are retained. The only modification is that each subband is extended down to DC. By this means, $f_{min}(s_k) < f_{min}(x)$ is always satisfied and a good fit of s_k is identified by testing each s_k in the range $k=K$-1..0 until a subband that satisfies $f_{max}(s_k) > f_{max}(x)$ is found. A problem is that no account is taken of $f_{min}(x)$ which is desirable in order to arrive at the best approximation to the optimal sample rate $f_{opt}(x)$ via eqn. (1.4). For high values of $Q(x)$, this can result in excessive oversampling redundancy. Due to the fact that $f_{min}(s_k) = DC$ rather than $f_{min}(s_k) = f_{max}(s_k)/2$, the fully-overlapping series has sample rates twice as high as the equivalent non-overlapping series in Fig. 3.3.

One option for reducing this overhead is to introduce a margin of overlap using a filterbank structure discussed in section 5.3. which permits arbitrary frequency displacements of subbands in the audio spectrum (Phillips et al, 1994). The amount of

overlap is related to an empirical assumption about the lowest value of $Q(x)$ permissible for most AS applications. Fig. 3.4 also illustrates a partially-overlapping subband series formed by setting $f_{min}(s_k)= (f_{min}(s_{k+1})+f_{max}(s_{k+1}))/2$, that is mid-band of s_{k+1}, with $f_{min}(s_K)=DC$ and $f_{max}(s_0)=f_s/2$ while retaining an octave series in subband widths (and thus a set of $f_{opt}(x)$'s related by powers-of-2) for ease of scheduling. A significant handicap is that the constant-Q partial series of any note below a certain value of $Q(x)$ cannot be accommodated without some partials aliasing.

Fully-Overlapped

Partially-Overlapped
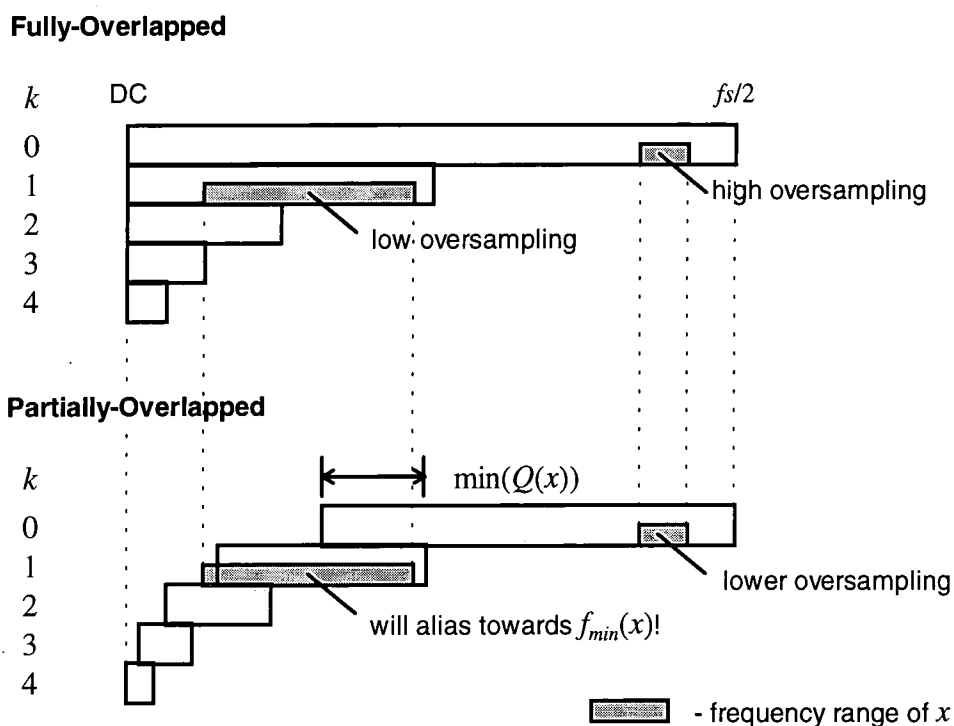
Figure 3.4   Fully and Partially Overlapping Octave-spaced Series for K=5

3.1.7 Towards an Objective Efficiency Comparison between AS and MAS

$$E = \frac{f_s u_{as} S}{v(K) + u_{mas} \sum_{k=1}^{K} f_s(s_k) n_k} \quad \text{where } S \equiv \sum_{k=1}^{K} n_k \qquad (3.2)$$

The general factors influencing the efficiency of MAS are summarised in eqn. (3.2) which represents the computational cost ratio E between AS via eqn. (1.1) and MAS: both for a single output stream $y[m]$. A successful MAS implementation should maximise E. The numerator is the net cost of a classical AS oscillator bank (e.g. the TOB architecture)

with oscillators updating at f_s at a unit cost of u_{as} per update Conversely, the denominator is the net cost of a MAS oscillator bank with costs $v(K)$ for filterbank computation (a function of K) and u_{mas} per oscillator update: n_k oscillators (S in total) are allocated to each s_k subband where $0 \leq k \leq K-1$. Note that u_{mas} is related to the distribution of the sample rates in s i.e. the set of $f_s(s_k)$'s. For instance, integer spacing in s has only a single sample rate implying low u_{mas}. In contrast, fully-overlapping octave spacing in s has K sample rates resulting in a higher multirate scheduling complexity and u_{mas}. The units for u_{as} and u_{mas} are implementation-dependent (e.g. CPU execution time or VLSI area) but have the same dimension.
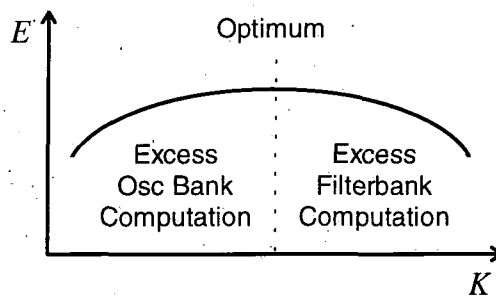
Figure 3.5   E versus v(K) given n

For a given scaleable filterbank scheme (e.g. fully-overlapping octave-spaced subbands), increasing K has the compound effect of increasing filterbank computation $v(K)$ and of providing a richer set of subbands in s. At a low value of K, filterbank computation is negligible, but a sparsely populated s results in the high probability of a poor fit of the selected 'best-fitting' subband to the typical $\{f_{min}(x), f_{max}(x)\}$ bounds of a oscillator x, and thus oversampling redundancy during synthesis. The extreme case is $K=1$ which is classical AS via eqn. (1.1). However, deploying excess filterbank resources with a high value of K yields an over-rich s that is poorly utilised as reflected by the resulting oscillator distribution n in eqn (3.2): the mean number of oscillators per subband will be low and, indeed, some may be empty. These trends are illustrated in Fig. 3.5. The maximum represents where K is optimised to the distribution of $\{f_{min}(x), f_{max}(x)\}$ bounds for the set of all S oscillators.

3.2 Proposal for a MAS Algorithm using QMF Filterbanks

Arising from the foregoing discussion, a subband decomposition exploiting the binary-tree structure of classical QMF (Quadrature Mirror Filter) filterbanks - with depth K - is identified which possesses the following desirable properties, facilitating a MAS algorithm which improves significantly upon the schemes discussed in sections 3.1.5. to 3.1.6:-

- No constraints are imposed upon $Q(x)$; even classical AS is included in the algorithm.

- Cost of note computation is optimised to $Q(x)$; fixed pitch notes are cheaper to compute (via integer-spaced subbands) than pitch-modulated notes (via octave-spaced subbands).

- Low computational cost (optimised through extensive research) and a simple scaleable filterbank topology using instantiations of a single protoype stage.

- A small set of $K+1$ subband sample rates (where all ratios are expressed in integer powers-of-two) reduces multirate oscillator bank and filterbank scheduling.

- A rich set of subbands (numbering 2^K-1) with which to guarantee "goodness-of-fit" of subband to the $\{f_{min}(x), f_{max}(x)\}$ bounds of an arbitrary oscillator x.

3.2.1 QMF Filterbanks

QMF Filterbanks are based upon the idealised prototype stage illustrated in Fig. 3.6 (Vaidyanathan, 1993). $H0(z)$ is an interpolating low-pass filter of $I=\uparrow 2$, known as a *half-band filter* because the cutoff frequency $z=\pi/2$ is at half the Nyquist frequency $z=\pi$. Observe that the stopband of $H0(z)$ suppresses the unwanted image of the baseband spectrum of $X0(z^2)$ outside $-\pi/2 < z < \pi/2$ (use of z^2 indicates that z is with reference to the $y[m]$ sample rate). In conventional $I=\uparrow 2$ interpolation, this band contains no energy in the output spectrum $Y(z)$. However, the QMF stage accepts a second signal for interpolation, $X1(z^2)$, which is filtered by $H1(z)$, a high pass filter, that passes the image and suppresses the baseband. When added to the signal from $H0(z)$, this image fits in the "gap" outside $-\pi/2 < z < \pi/2$. Note that the image is inverted in frequency. The term 'Quadrature Mirror' derives from the fact that $H1(z)$ is the mirror image of $H0(z)$ in

$z=\pi/2$, a property which economises on computation (as discussed in Chapter 4). An important conceptualisation of the QMF stage is as a node of a binary tree that divides its output band $y[m]$ into two equal-width, non-overlapping and contiguous subbands $x0[m]$ and $x1[n]$. The implications of non-ideal filters are dealt with in later chapters and, for the moment, it is preferable to discuss the ideal case.

$$X0(z^2)$$

$$H0(z) \text{ (Low Pass)}$$

$$X1(z^2)$$

$$H1(z) \text{ (High Pass)}$$

$$Y(z)=X0(z^2)H0(z)+ X1(z^2)H1(z)$$
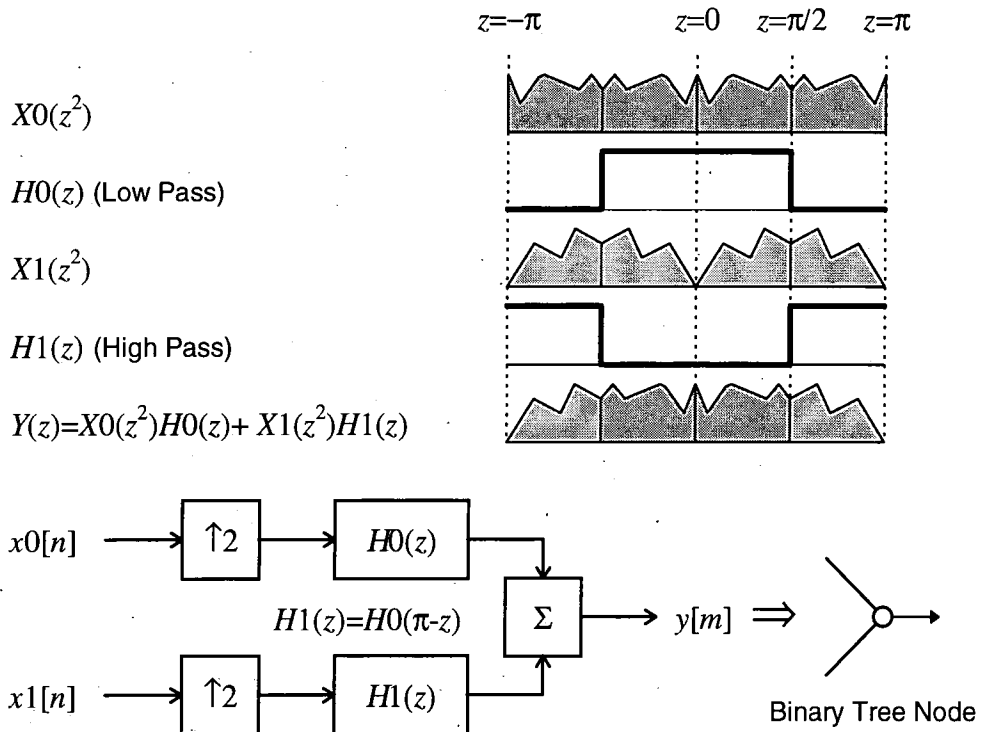
Figure 3.6   Operation of QMF Synthesis Filterbank Stage

3.2.2 QMF Subband Hierarchy

A QMF filterbank is created by assembling instantiations of the prototype stage of Fig. 3.6 into a binary tree of arbitrary completeness: 'complete' means that the number of stages on any path from the root fullband to any terminating subband is always K. An example is illustrated in Fig. 3.7 for $K=3$. A constraint for MAS is that worst-case latency from level K exponentiates with K: a low value is necessary for real-time synthesis within T_{max}. As justified in section 6.4.4, an optimal value appears to be $K=3$ which is the default value used in subsequent discussion. In spite of a ceiling on K, configurability of filterbank topology permits $v(K)$ in eqn. (3.2) to be optimised, given knowledge of the $\{f_{min}(x), f_{max}(x)\}$ distribution of the oscillator set, to maximise E. From

the perspective of MAS, a critical feature of a QMF filterbank is that it can be interpreted as a *subband hierarchy* when intermediate subbands between QMF stages are considered. These are usually ignored in conventional analysis / synthesis applications because QMF filterbanks are perceived exclusively as an efficient way to generate a series of 2^K integer-spaced subbands at level K: intermediate signals are of no interest (Crochiere and Rabiner, 1983). Fig. 3.7 illustrates both a QMF filterbank and its corresponding subband hierarchy.

QMF Binary-Tree of Depth $K=3$

$s'_{k,l}$ denotes frequency inversion
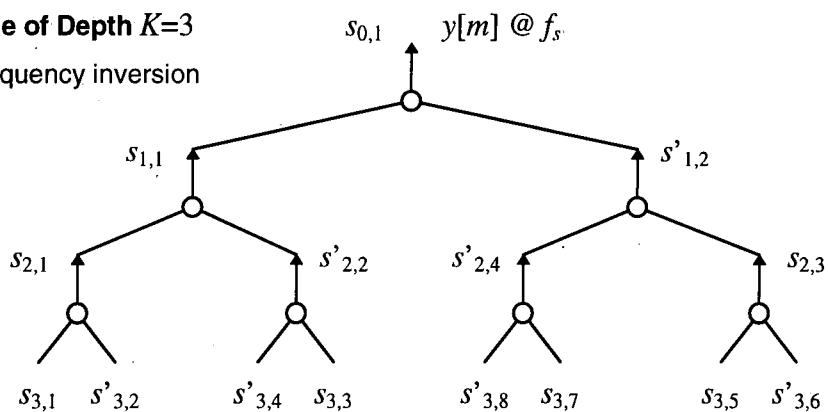
Normalised Sub-band Hierarchy

Level k	DC							$f_s/2$
0	$s_{0,1}$							
1	$s_{1,1}$				$s_{1,2}$			
2	$s_{2,1}$		$s_{2,2}$		$s_{2,3}$		$s_{3,4}$	
3	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$	$s_{3,4}$	$s_{3,5}$	$s_{3,6}$	$s_{3,7}$	$s_{3,8}$

Figure 3.7   Example QMF Filterbank and Subband Hierarchy for K=3

Each subband in the hierarchy is denoted by $s_{k,l}$:$\{0\leq k\leq K,\ 1\leq l\leq 2^k\}$. l indexes the 2^k integer-spaced subbands at level k. The mapping from tree to hierarchy is indirect because, in an individual QMF stage, an inverted side-band image of the baseband of $x1[n]$ appears in the output $y[m]$, and therefore if $x1[n]$ subdivides into subbands itself, these are 'flipped' over in frequency domain. To generalise, if the filterbank path from a subband $s_{k,l}$ to the filterbank output includes an odd number of $H1(z)$ filters, $s_{k,l}$ is

inverted, and if the number is even, $s_{k,l}$ is non-inverted. Inverted subbands are denoted by $s'_{k,l}$ in Fig. 3.7. This peculiarity is easily compensated for in the proposed MAS algorithm by pre-inverting the $F_x[n]$ frequency envelope of an oscillator x allocated to an $s'_{k,l}$ (see section 4.2.5). However, it is convenient to consider a 'normalised' subband hierarchy, whilst acknowledging the indirect mapping to a non-normalised topology.

The internal structure of the subband hierarchy has many desirable properties. Firstly, there are K integer-spaced subband series from the terminal series at level K up to the fullband which is classical AS via eqn. (1.1). Secondly, the series constituted by $s_{3,1}$ and $s_{k,2}:\{1 \leq k \leq K\}$ is a non-overlapping octave-spaced series whereas $s_{k,1}:\{0 \leq k \leq K\}$ is the contrasting fully-overlapping equivalent. Therefore the subband decompositions of sections 3.1.5 and 3.1.6 are included in the hierarchy. Conceptually, if a note partial series does not fall into one of the allocation patterns presumed by one of these formal decompositions, others are available. The necessity for partially-overlapping subbands is avoided. Another advantage is that intermediate subbands are interpolated 'for free' in that the higher QMF stages are necessary structurally to recombine the terminal subbands. The computational cost of a complete tree is Kc (where c is the cost of the final stage executing at f_s) because at each extra level, though the number of stages doubles, the sample rate is halved, reflecting the principle of the multistage interpolator in section 3.1.2. Furthermore, a polyphase decomposition of the prototype QMF stage minimises computation as discussed in Chapter 4.

3.2.3 Allocation of $\{f_{min}(x), f_{max}(x)\}$ in a Subband Hierarchy

An allocation algorithm for determining the optimal subband $s_{k,l}$ from a subband hierarchy for an arbitrary oscillator x - bounded by $\{f_{min}(x), f_{max}(x)\}$ - must use the criteria of section 3.1.3 i.e. identify the minimum $f_s(s_{k,l})$ that satisfies $f_{max}(s_{k,l}) > f_{max}(x)$ and $f_{min}(s_{k,l}) < f_{min}(x)$. A conceptual refinement is to identify a level k such that subband width has the 'best-fit' - the minimum k that satisfies $f_s 2^{k-1} > (f_{max}(x) - f_{min}(x))$ - and test whether $\{f_{min}(x), f_{max}(x)\}$ is contained by a single subband. If so, then x can be allocated. If, however, $\{f_{min}(x), f_{max}(x)\}$ spans one of the vertical discontinuities in the hierarchy, and therefore risks aliasing, it is 'promoted' to the sub-band immediately above the termination of the discontinuity. In the extreme case of $f_{min}(x) < f_s/4 < f_{max}(x)$, promotion is

to the computationally most expensive top band $s_{0,1}$ which is classical AS. However, observe that in Fig. 3.7 discontinuities which promote to high subbands are few, and that those with smaller promotion distances are numerous. For the optimal level k, then, of its 2^k-1 discontinuities, the number promoting to the next level is 2^{k-1} i.e. over 50%.
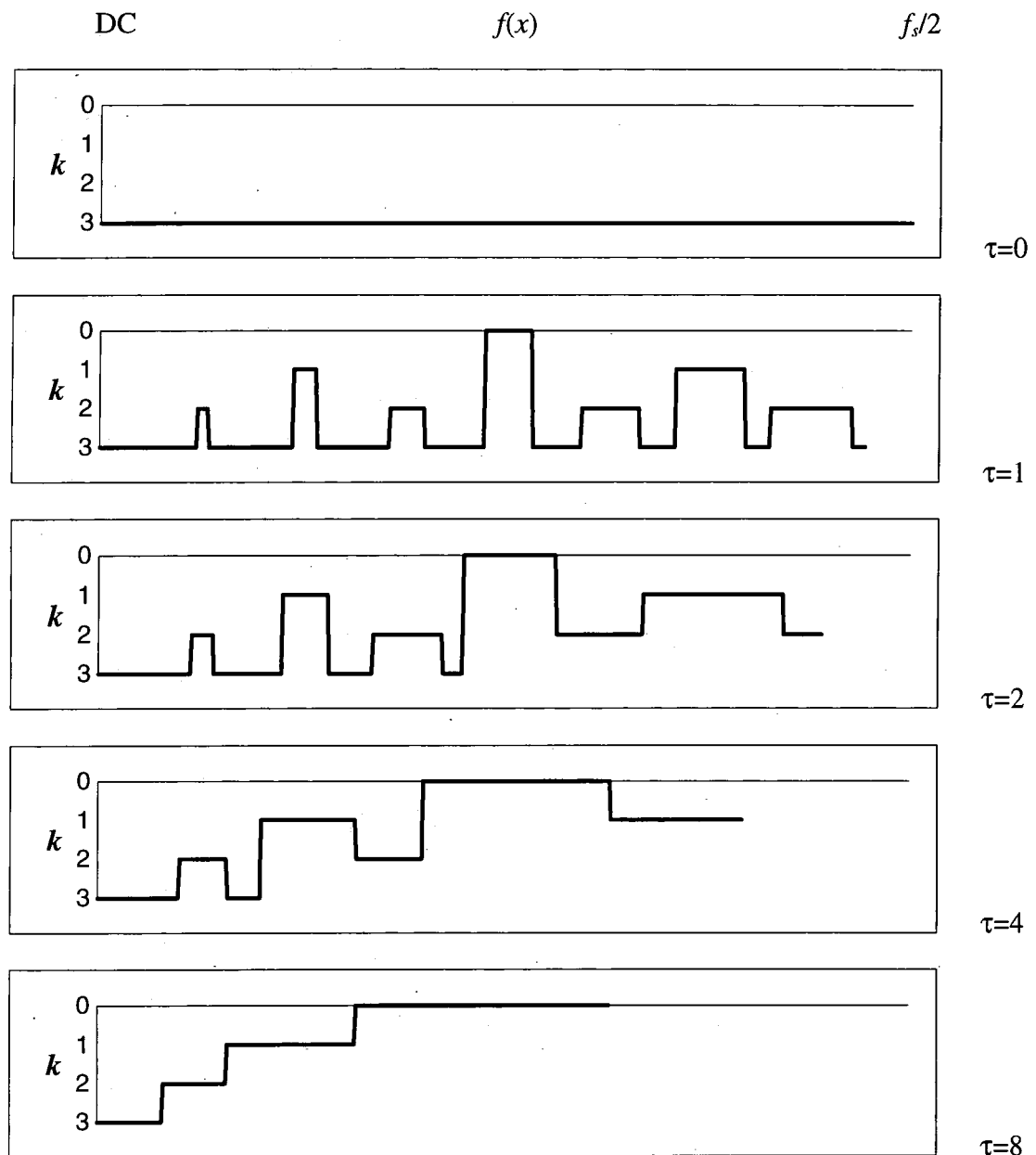
Figure 3.8   Evolution of Subband Hierarchy Allocation Pattern with Increasing τ

A hypothetical constant-Q scenario from section 3.1.4, demonstrating the advantage of a subband hierarchy, is to consider the allocation pattern of a note with partials in pitch-invariant harmony undergoing symmetrical modulation of $\pm\tau$ semitones from its centre

pitch. If partial x has centre frequency $f(x)$ then $f_{min}(x)=f(x)/\delta_f$ and $f_{max}(x)=f(x)\delta_f$ where $\delta_f=2^{\tau/12}$. Fig. 3.8 illustrates the evolution of the pattern *versus* $\tau \in \{0,1,2,4,8\}$ where the abscissa is $f(x)$, and the ordinate is the allocation level of x in the hierarchy of Fig. 3.7. The resulting allocation patterns are independent of note pitch and a function of τ alone, serving as a template for the actual partial series. At $\tau=0$, partials have a very high-Q and are allocated to the deepest integer-spaced series $s_{3,1..8}$. As τ increases, the pattern gradually evolves into the fully-overlapping octave-spaced series $s_{0..3,1}$ that is suitable for low-Q partials. Evidence of computational efficiency is the mean subband sample rate expressed as a factor υ relative to classical AS at f_s in $s_{0,1}$ as a function of τ which is plotted in Fig. 3.9: theoretical υ is artificially high because it assumes a complete set of high treble partials up to the Nyquist limit $f_{max}(x) \leq f_s/2$. In practice, this band is likely to be sparsely occupied (Sandell, 1994) and thus practical results will produce a lower υ. Two properties of subband hierarchy are thus confirmed in that (i) cost of note computation is optimised to the expected value of τ and (ii), no constraints are placed on τ.
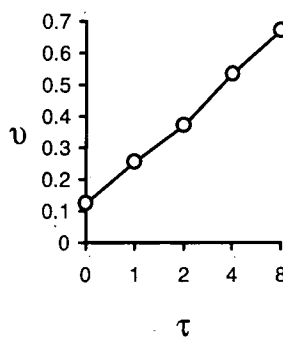
Figure 3.9 υ versus τ

3.3 A Comparison of AS via IFFT and Multirate Techniques

FFT^{-1} is a block transformation resulting in a coarse frame-rate control resolution, permissible because of the comparable coarse resolution of the human ear to envelope features. However, maintenance of the constant-Q principle is desirable, in order to reflect the properties of natural sounds. MAS supports constant-Q allocation in octave-

spaced subbands and facilitates a graduated tradeoff towards integer spacing, analogous to the single control rate of FFT^{-1}. Via a hierarchical subband decomposition, MAS quantises oscillator frequency range, but it is a logical reflection of the stationary nature of note pitch, and of a partial series bound to that pitch. FFT^{-1} does not require $\{f_{min}(x),$ $f_{max}(x)\}$ of each oscillator x, but it is asserted that, for note-based music, such *a priori* information is available by default and therefore worthy of exploitation. An analogy is often drawn between IFFT / OLA synthesis with N frequency bins and an integer-spaced multirate filterbank with N subbands (Vaidyanathan, 1993). They represent complementary frequency-domain and time-domain approaches to optimising the tradeoff between Δ_t and Δ_f in Heisenberg's inequality of eqn. (1.5).

FFT^{-1} has an advantage over MAS for noise synthesis as the latter is dedicated to the optimal computation of sinusoids. However, closely spaced sinusoids randomly modulated in amplitude and frequency are capable of providing a satisfactory alternative to filtered noise in many circumstances (Jansen, 1991). Indeed, there is benefit to be gained from an integration of noise and sinusoids in the same model as, often, the distinction is artificial. For instance, the rapid decay of high frequencies in an attack transient is perceived as a short non-tonal noise. In contrast, SMS extracts a smoothed spectral envelope for noise and requires signal stationarity for partial classification (Masri and Bateman, 1996). Randomisation of envelopes is incompatible with PWL modelling which presupposes slow rates of control parameter change. A MAS oscillator bank may benefit from extensions to facilitate envelope randomisation using algorithms such as that proposed by Fitz and Haken (1995). 'Noisy' oscillators are likely to obey the constant-Q principle and therefore be compatible with allocation in a subband hierarchy.

4. Practical QMF Filterbank Design

4.1 Overview

The ideal QMF filters ($H0(z)$ and $H1(z)$) in section 3.2.1 consist of perfect rectangular frequency domain windows and consequently have infinite latency, order and computational cost. This assumption has utility in exploring the general properties of subband hierarchy but for a practical implementation, major compromises are necessary. A primary limitation is that latency for real-time operation, as specified in section 1.2.2, has an upper bound of $T_{max} \cong 20$ms. Additionally, transferring the computational burden from oscillator bank to filterbank has the risk, in the extreme, of resulting in worse efficiency than classical AS (see section 3.1.7). Another impetus for reducing prototype filterbank complexity is to maximise the number of filterbank instantiations that may be multiplexed through the same hardware in order to provide a satisfactory number of independent output streams as discussed in section 1.2.1. A related topic, discussed in section 4.2 onwards, is in the modifications required for functional transparency.

The initial topic of this chapter is an investigation of efficient standard QMF filterbank designs in the context of their application to MAS. Both FIR and IIR forms are possible and each has attendant advantages and disadvantages. Practical QMF stage design is parameterised by the classical interpolator design variables from section 3.1.1. which are Δ_f (transition width) and δ_p and δ_s (passband and stopband ripple). It is appropriate to set out their tolerances prior to documenting their impact on QMF design. δ_s controls the relative power of quantisation noise transmitted during interpolation and, ultimately, interpolated oscillator SNR; for high-fidelity synthesis $\delta_s = -80$dB which is comparable to that used in TOB's of recent years (Jansen, 1991). The constraint upon δ_p is quite subjective; an oscillator placed at a passband ripple maximum should have the same perceived power as at a minimum. However, given δ_s, varying δ_p has little effect on filterbank cost. Therefore the single design variable of interest is Δ_f.

4.2 FIR QMF Filterbank Design

4.2.1 Optimal Structure for a FIR QMF Stage

Finite Impulse Response (FIR) digital filter designs for QMF stages are well documented in the literature (Crochiere and Rabiner, 1993) and hence a brief tutorial will suffice. Invariably, symmetrical impulse responses are used which have a latency of precisely M samples where the filter length is $N=2M+1$. Applying an inverse Fourier transform to the ideal rectangular "brick wall" frequency response of $H0(z)$, which is unity within the bounds $\pm\pi/2$ and zero elsewhere creates the well-known time series for $h0[m]$ described by eqn. (4.1), known as a *sinc* function. $H1(z)$ is $H0(z)$ displaced in frequency by $f_s/4$ which has the time series $(-1)^m$. When multiplied by $h0[m]$, the desired impulse response for $h1[m]$ is obtained in eqn. (4.2). m is bounded such $M\leq m\leq M$ and M is odd so that coefficients at the extremes are non-zero to eliminate redundant latency. Example functions are plotted in Fig. 4.1 for $M=11$ ($N=23$): note that $h0[m]$ and $h1[m]$ are non-zero for all odd m. Truncation of the infinite time series gives rise to non-ideal filter behaviour which can be minimised by the techniques discussed in the next section.

$$h0[m] = 2\left(\frac{\sin(m\pi/2)}{m\pi}\right)$$

$$h1[m] = (-1)^m h0[m]$$

$$(4.1, 4.2)$$
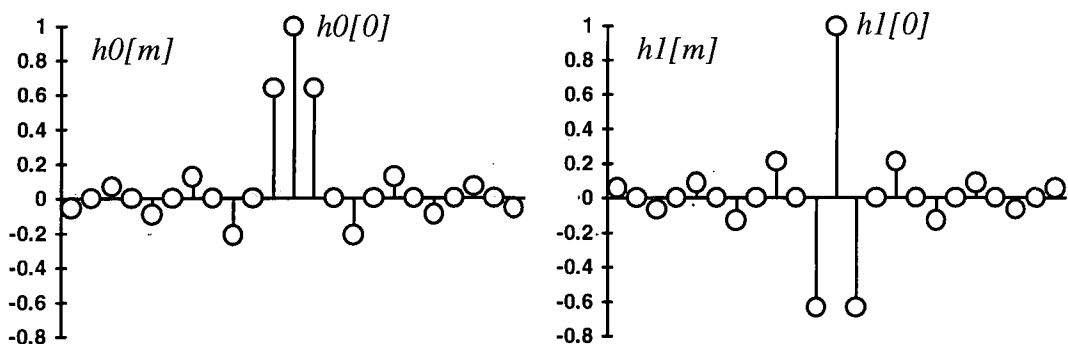
Figure 4.1   *Unwindowed FIR Responses for h0[m] and h1[m] for M=11, N=23*

4.2.2 FIR Frequency Response Optimisation

A truncated *sinc* function for $H0(z)$ is equivalent to a rectangular window which has poor frequency domain performance. A solution is to employ the Kaiser window FIR design technique which minimises filter length by shaping the frequency response such that it is closely bounded by the tolerance margins Δ_f and $\delta = \delta_s = \delta_p$ from which M is determined by eqn. (4.3). The formulae for the time-domain window is based on a zeroth-order Bessel function and is in the literature (Crochiere and Rabiner, 1993). Note that the same ripple is specified for passband and stopband. Kaiser filters are sub-optimal because, in practice, the height of the first stopband sidelobe is closest to δ and the rest fall monotonically below this value, constituting excess stopband gain (see Fig. 4.3). However, it is simple to compute and hence a popular design technique where strict optimality is not required.

$$M = \frac{-20\log_{10}\delta - 7.95}{28.72\Delta_f} \qquad (4.3)$$

An alternative to windowing is the Parks and McClellan (1972) "equiripple" design algorithm. This takes the more specific parameters of $\{f_{pass}, f_{stop}, \delta_p, \delta_s\}$ and applies an iterative technique, the Remez exchange algorithm, to adjust the sidelobe amplitudes in the passband and stopband to conform to the equiripple condition (i.e. of equal height). For a given specification, it achieves a design with minimum M. Empirical formulae for M are given in eqns. (4.4, 4.5, 4.6) which are quoted from (Crochiere and Rabiner, 1993). It differs from the Kaiser technique in that the condition $\delta_p = \delta_s$ is not necessary. Specifying $\delta_p \neq \delta_s$ results in lower M (and therefore lower latency) but also generates non-zero coefficients at even-numbered values of m making it more expensive to compute than a windowed *sinc* design. However, setting $\delta_p = \delta_s$ and $f_{pass} = \pi - f_{stop}$ results in an optimally windowed *sinc* function with zero coefficients at even-numbered m, of less M than the Kaiser window. Fig. 4.3. provides a visual comparison of the performance of the two FIR design methods for a typical MAS specification as indicated by section 6.4.4.

$$D_\infty(\delta_p, \delta_s) = \log_{10}(\delta_s(0.005309(\log_{10}\delta_p)^2 + 0.07114\log_{10}\delta_p - 0.4761))$$
$$+(-0.00266(\log_{10}\delta_p)^2 - 0.5941\log_{10}\delta_p - 0.4278)$$

(4.4)

$$f(\delta_p, \delta_s) = 11.012 + 0.512(\log_{10}\delta_p - \log_{10}\delta_s), \text{ for } |\delta_s| \le |\delta_p|$$

(4.5)

$$M = \frac{D_\infty(\delta_p, \delta_s)}{2\Delta_f} - \frac{f(\delta_p, \delta_s)\Delta_f}{2}$$

(4.6)

Kaiser Window M=25, N=51

Parks-McClellan M=23, N=47

Gain (dB) vs Frequency (Nyquist=1.0)   Gain (dB) vs Frequency (Nyquist=1.0)
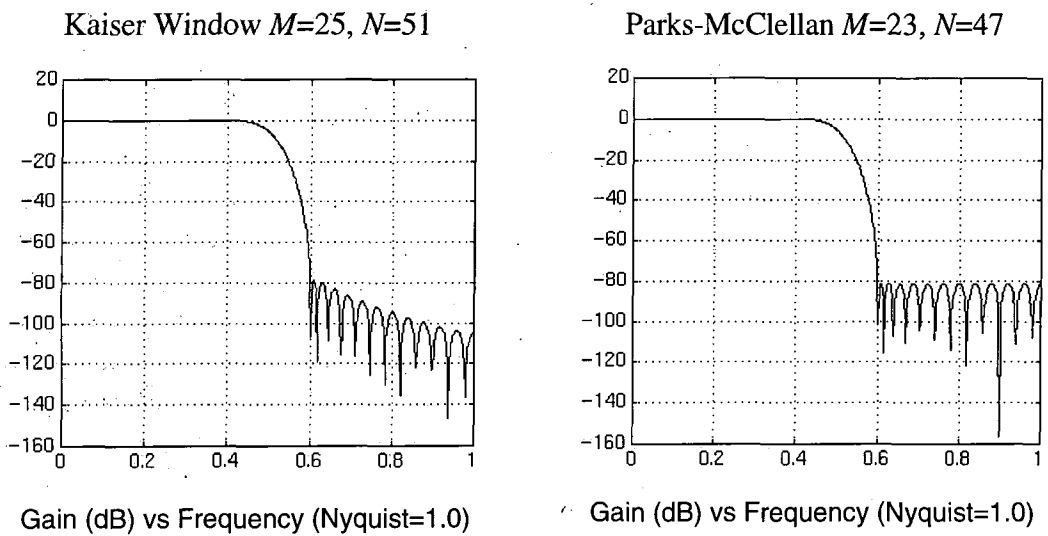
Figure 4.3   *FIR designs for H0(z) with Δ_f=0.1, δ_p=δ_s=-80dB*

4.3 MAS Normalisation Techniques for QMF FIR Filterbanks

The objective of achieving functional transparency requires consideration. For MAS to integrate with existing spectral modelling techniques, it should appear to the high-level SME, and ultimately the musician, as a TOB (granted the need for *a priori* data). However, fragmentation of a note partial series across different subbands in a subband hierarchy corrupts (i) time, (ii) frequency and (iii) phase relationships between partials. Normalisation techniques which address each of these problems in an FIR QMF filterbank are presented in the following sub-sections, and form the basis of those used in the IIR QMF implementation of choice, discussed in section 4.4.

4.3.1 Latency Normalisation

$$T_{fb}(K) = \frac{M}{fs} \sum_{i=1}^{K-1} 2^{i-1} = \frac{M(2^{K-1}-1)}{fs} \qquad (4.7)$$

Assume the case of a note partial series allocated across a constant-Q series of subbands as illustrated in section 3.2.3. High frequency partials in the fullband (k=0) are not subject to any filterbank delay. Lower frequency partials are allocated deeper in the hierarchy and for allocation level k are subject to a latency of $T_{fb}(k)$ according to eqn. (4.7) which exponentiates with k. Therefore at note onset, high frequencies emerge before low frequencies. Though this phenomena is a reflection of some natural processes, it is desirable to align envelope features between partials for accurate resynthesis: feature relationships in PWL envelope data encapsulate the spectral characteristics of the analysed source tone and should remain undistorted by the filterbank. A simple solution is to include delay lines for inputs to subbands k<K such that latency from any level in the filterbank is normalised to the maximum of $T_{fb}(K)$. This is possible because FIR QMF stages exhibit a frequency-independent group delay of M samples. The length of delay line $d(K,k)$ required for level k is expressed by eqn. (4.8). The deepest subbands at level K do not require delay-lines.

$$d(K,k) = M(2^{K-k}-1) \text{ for } k < K \qquad (4.8)$$

4.3.2 Frequency Normalisation

Once an oscillator x is allocated to the optimal subband $s_{k,l}$, its PWL frequency envelope $F_x[m]$ - from eqn. (1.1) - requires translation into the context of $s_{k,l}$ such that the desired frequency appears at the filterbank output. This is achieved via the linear relationship of eqn. (4.9) which generates the required digital frequency envelope $\Omega_x[m]$. If $s_{k,l}$ is inverted in frequency (by interpolation via an odd number of H1's, see section 3.2.2), $\Omega_x[m]$ must be pre-inverted to cancel the effect. The computational overhead in a PWL context is trivial as only normalisation of breakpoints is necessary, and the divisor for

$$\Phi_x[n] = \Phi_x[n-1] + 2^{-k} D\left(\frac{a + \Omega_x[n]}{2}\right) + \phi'_x[n]$$

$$\text{where } a = \Omega_x[n] + \frac{\Omega_x[n] - \Omega_x[n-1]}{D}$$

(4.10)

4.3.3 Phase Normalisation

At the output of the filterbank, it is desirable that each sinusoid should have correct relative phase: a property usefully termed as "phase transparency". To analyse the phase characteristics of the QMF FIR synthesis filterbank, consider the properties of a single stage. A stationary sinusoid is applied to each input, both lowpass ($x0[n]$) and highpass ($x1[n]$) according to eqns. (4.11, 4.12). At $n=0$, $x0[n]$ and $x1[n]$ have instantaneous phase ϕ_0 and ϕ_1 respectively. After a delay of M samples at the output, the interpolated baseband of $x0[n]$ preserves its phase of ϕ_0 at $y[m=M]$ as expressed in eqn. (4.13). The only transformation is a halving of relative digital frequency Ω_0 independent of ϕ_0.

$$x0[n] = A_0 \sin(\Omega_0 n + \phi_0)$$
$$x1[n] = A_1 \sin(\Omega_1 n + \phi_1)$$

(4.11, 4.12)

$$y[m - M] = A_0 \sin(\frac{\Omega_0}{2} m + \phi_0) + A_1(\sin\frac{2\pi - \Omega_1}{2} m + \frac{2\pi - \Omega_1}{\Omega_1} \phi_1)$$

(4.13)

However, in the context of the output sample rate, highpass filter $H1$ suppresses the baseband signal ($\Omega_1/2$) of $x1[n]$ and passes the sideband at $\pi - (\Omega_1/2)$. As phase is the integral of frequency, the phase of the $x1[n]$ sideband at $y[m=M]$ (the product of the baseband phase ϕ_1 and the frequency ratio of sideband to baseband) is distorted from its desired value ϕ_1 and is, furthermore, a function of oscillator frequency Ω_1. However, substituting a frequency-dependent phase factor in eqn. (4.12) to produce eqn. (4.14) cancels out this phase distorting term in eqn. (4.13) such that the interpolated sideband of $x1[n]$ has the desired phase ϕ_1 at $y[m=M]$.

$$x1[n] = A_1 \sin(\Omega_1 n + \frac{\Omega_1}{2\pi - \Omega_1} \phi_1)$$

(4.14)

$$\phi'_x[n] = c[n]\phi_x[n] \text{ where } c[n] = \frac{\text{base - band freq}}{\text{side - band freq}} = \frac{f_s 2^{-k}\Omega_x[n]}{2\pi F_x[m]}$$

<div align="right">(4.15)</div>

where $m = nD + d(K,0)$ via eqn.(4.8) and $D = 2^k$ as in section 4.3.2

This principle may be extended to a QMF filterbank (e.g. the complete binary tree of Fig. 3.7) which includes delay lines for latency normalisation. According to eqn. (4.8), latency is a constant $d(K,0)$ samples irrespective of allocation depth and therefore phase relationships are preserved during interpolation. Phase normalisation can therefore be interpreted as the adjustment of baseband phase to give the required sideband phase. An oscillator x allocated to subband $s_{k,l}$ has actual baseband frequency $f_s 2^{-k}(\Omega_x[n]/2\pi)$ via eqn. (4.9) which, after suppression of unwanted side-bands in the filterbank, corresponds to the desired frequency of sideband $F_x[m]$ at the output. To ensure $F_x[m]$ has the desired phase $\phi_x[m]$ requires substitution of the normalising eqn. (4.15) in eqn. (4.10).

For a stationary sinusoid, eqn (4.15) need only be computed once. However, for sinusoids in AS with time-varying frequency, eqn. (4.15) must be computed each sample period to maintain precise phase relationships. As the computation involves a division, this constitutes a large potential overhead. It is redundant, however, in the case of the subband series $s_{k,l}$:{$k=0..K,l=1$} which comprises a cascade of $H0$ lowpass filters where the phase normalising term in eqn. (4.5) is unity: the baseband image itself is passed by the filterbank. This cascade is functionally equivalent to the fully-overlapping octave spaced subband decomposition discussed in section 3.1.6 and is an advantageous property of the FIR QMF filterbank (see section 4.5).

4.4 IIR QMF Design

Restricting the subject of Infinite Impulse Response (IIR) filter theory to that of QMF stage design singles out the family of *polyphase all-pass* (PA-IIR) filters that have low computational cost, and narrow tolerances for Δf, δ_s and δ_p. The building block of a PA-IIR QMF is a single stage allpass filter (Oppenheim and Schafer, 1989), which has a unity gain, but non-linear phase response governed by a single coefficient α (the z-transform consists of a pole at $-1/\alpha$ and a zero at $-\alpha$) with a difference equation of

$y[n]=\alpha(x[n]-y[n-2])+x[n-2]$. Prime advantages are that it is cheap to compute, unconditionally stable, and because it has unity gain, may exploit the full dynamic range of fixed-point arithmetic (in contrast to other IIR building-blocks such as 2nd order sections). The phase response of a cascade of n all-pass filters with response $G(z)$ is the sum of the individual phases dictated by α_i: $\{1 \leq i \leq n\}$ as expressed in eqn. (4.16).

$$\angle G(z) = \sum_{i=1}^{n} \angle \left(\frac{\alpha_i + z^{-1}}{1 + \alpha_i z^{-1}} \right)$$
(4.16)

Fig. 4.5. illustrates the phase response of a state-of-the-art high-performance IIR polyphase allpass filter as published by (Hart et al, 1993); an example which is most suitable for closer investigation. Two parallel all-pass cascades are used; $G0$ and $G1$ ($G1$ is in series with a unit delay z^{-1}). $G0$ has length $n=6$ with $\alpha_i \in \{0.984964, 0.798278, 0.914901, 0.593341, 0.297311, 0.040409\}$ and $G1$ has length $n=5$ with $\alpha_i \in \{0.865132, 0.953132, 0.708912, 0.452729, 0.149350\}$. Coefficients are derived by an iterative design procedure. A signal is input into both cascades and at the output it can be seen that the relative phase difference (bold line) is zero for $0 \leq z \leq \pi/2$ (i.e. $G0$ and $G1$ in-phase) and $-\pi$ radians for $\pi/2 \leq z \leq \pi$ (i.e. $G0$ and $G1$ in antiphase) with a narrow transition width. Clearly, summing the output creates a half-band filter with gain $\times 2$ for $0 \leq z \leq \pi/2$ and $\times \equiv 0$ for $\pi/2 \leq z \leq \pi$, equivalent to the $H0$ filter in a QMF stage.
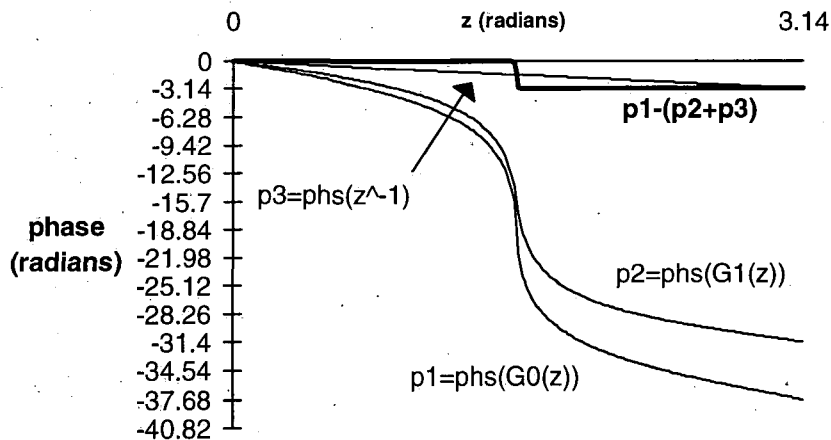
Figure 4.5   Phase Response of PA-IIR Filter

To translate $G0$, $G1$ into an equivalent QMF synthesis stage is straightforward. However for a polyphase implementation equivalent to the FIR QMF of Fig. 4.2, filters with the response $G0(z^2)$ and $G1(z^2)$ are required. This is achieved by decimating the difference equation by $\downarrow2$ to $y[n]=\alpha(x[n]-y[n-1])+x[n-1]$ which permits filter computation at the lower sample rate with a net cost of 5.5 multiplies per $y[m]$. The commutation sequence implements the required z^{-1} series delay for $G1$. Fig. 4.6 illustrates the resulting structure. Half-band interpolation of the low-pass signal $x0[n]$ is performed as previously described giving the standard QMF response $H0(z)$. Interpolation of the high-pass signal $x1[n]$ by QMF response $H1(z)$ is realised by sending an inverted copy of $x1[n]$ to $G1$. This causes antiphase cancellation of the outputs of $G0$, $G1$ over $0\leq z\leq\pi/2$ and in-phase superposition over $\pi/2\leq z\leq\pi$ and therefore realises $H1(z)$. Gain ($\times2$) from superposition is normalised to unity at the output. The actual normalised frequency response is illustrated in Fig. 4.7: note that $\delta_s =-100$dB which is an improvement of 20dB over the specification in section 4.1 of $\delta_s=-80$dB.
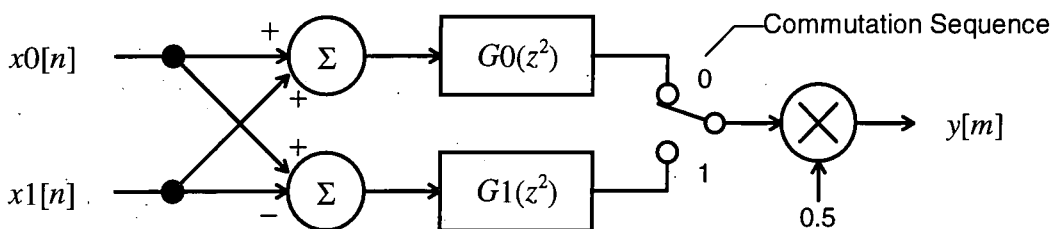
Figure 4.6   IIR Polyphase Allpass QMF Synthesis Stage
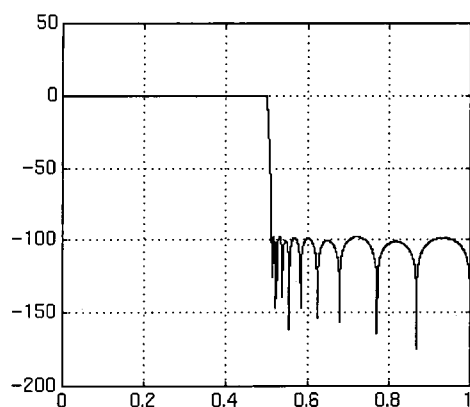
Gain (dB) vs. Frequency (Nyquist==1.0)

Figure 4.7   PA-IIR H0(z) Response

4.4.1 Normalisation with Non-linear Phase

Frequency normalisation in a QMF filterbank is independent of its phase response and hence the method in section 4.3.2 also applies to the PA-IIR design. However, the group delay of the PA-IIR stage is frequency-dependent in contrast to the fixed delay of M samples in a symmetric FIR filter. This means that phase relationships are distorted and envelope features are skewed by a PA-IIR filterbank during interpolation. To normalise phase therefore, requires the inclusion of the filterbank transfer function in eqn. (4.15). Latency normalisation by fixed-length delay lines (as proposed for the FIR QMF in section 4.3.1) is not possible as they function in the context of linear phase alone.

An interesting software alternative is to compute the latency T_x for a high-Q partial x at frequency $f(x)$ from the transfer function and "pre-skew" the $A_x[n]$, $F_x[n]$ envelopes by $T_{fb}(K)$-T_x, where $T_{fb}(K)$ is the PA-IIR filterbank worst-case latency. In a PWL envelope context, this may be done by attaching horizontal segments of length T_x to the start of the $A_x[n]$, $F_x[n]$ envelopes at note onset where T_x is computed from an LUT. However, the actual delay of a PA-IIR QMF stage is small, relative to the example PM-FIR QMF's discussed in section 4.2.2. As illustrated by Fig. 4.5 the greatest delay is -5π radians at $z=\pi/2$: that is about 10 samples.
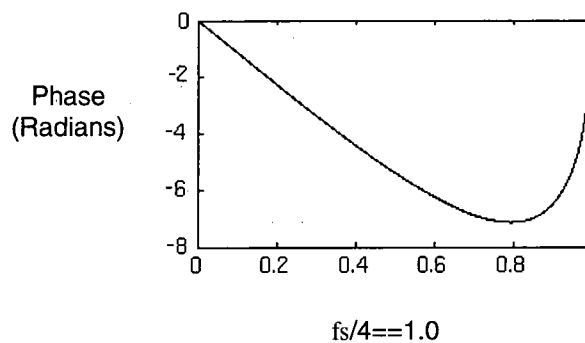
Figure 4.8   Required Passband Phase Response of P(z)

A structural solution to both of these problems is to incorporate an allpass phase-equaliser $P(z)$ ($|P(z)|=1$ for $\forall z$) into the prototype PA-IIR QMF stage output such that its net phase response from input to output is approximately linear. It is then it is functionally interchangeable with an FIR QMF stage and the algorithms of sections 4.2.1

and 4.2.2 are applicable. Modularity of the QMF filterbank structure is retained simplifying implementation. The alternative is to employ a single equaliser at the output, on the assumption that a single transfer equation characterises the entire filterbank.

Unfortunately, use of subband hierarchy in MAS means that at any particular value of absolute frequency, a signal can be sourced from one of a number of overlapping subbands, each with a unique transfer function and hence a single equaliser cannot be used. However, a problem with $P(z)$ is illustrated in Fig. 4.8 in that it has a phase response with a local minimum in the passband which cannot be realised by an allpass IIR filter which, by definition, has maximum phase. It is concluded that phase-linearity, and hence phase normalisation, is impractical for this PA-IIR QMF filterbank but that latency normalisation by envelope pre-skewing appears quite feasible and efficient.

4.5 Comparing PM-FIR and PA-IIR Performance for MAS

QMF Stage:	Generic FIR	PM-FIR from Fig. 3.5 (M=23)	PA-IIR
Storage (samples)	(M+1)	24	12
Mults per $y[m]$	(M+1)/4	6	5.5
Adds per $y[m]$	1+(M+1)/4	7	12

Table 4-1   Relative Complexity of FIR and PA-IIR QMF Stages

A comparison of the transition width characteristics between Figs. 4.3 and 4.7 demonstrates that the PA-IIR approach has superior performance: a feature that is a critical determinant in the choice of QMF filterbank for MAS as shall become clear in the next section. Additionally, δ_s=-100dB rather than δ_s=-80dB from section 4.1. The PM-FIR and PA-IIR QMF designs share the same general polyphase model (respectively Figs. 4.2 and 4.6) and differ only in the structure of their component phase shifters which may both function optimally in fixed-point arithmetic. Respective storage and computational requirements are tabulated in Table 4-1. Complexity of the PA-IIR stage

is thus comparable to the PM-FIR examples of Fig. 4.3. (which are good designs for typical MAS applications, see section 6.4.4)

If a non-linear phase response is tolerated, a PA-IIR implementation is most efficient in the light of its Δ_f performance. As discussed in section 4.3.3, phase normalisation in the PM-FIR QMF is complex to implement, but the subband series $s_{k,l}$:$\{k=0..K,l=1\}$ offers phase transparency by default for subbands where the majority of low-frequency oscillators will be allocated in MAS. Sensitivity of the ear is most critical up to around 5kHz and diminishes with increasing frequency (Parsons, 1987). Phase has particular relevance in the context of stereo and sound spatialisation, but such a criterion does not apply so strongly to MAS as the filterbank is accumulating monoaural sources for subsequent spatialisation by an unspecified post-processor (see section 8.3.1). In the general case, altering the relative partial phases in a musical tone has minimal impact upon the perception of its timbre (Plomp, 1976), but exceptions to the rule can be contrived.

4.6 On Non-infinitesimal Δ_f in a Subband Hierarchy
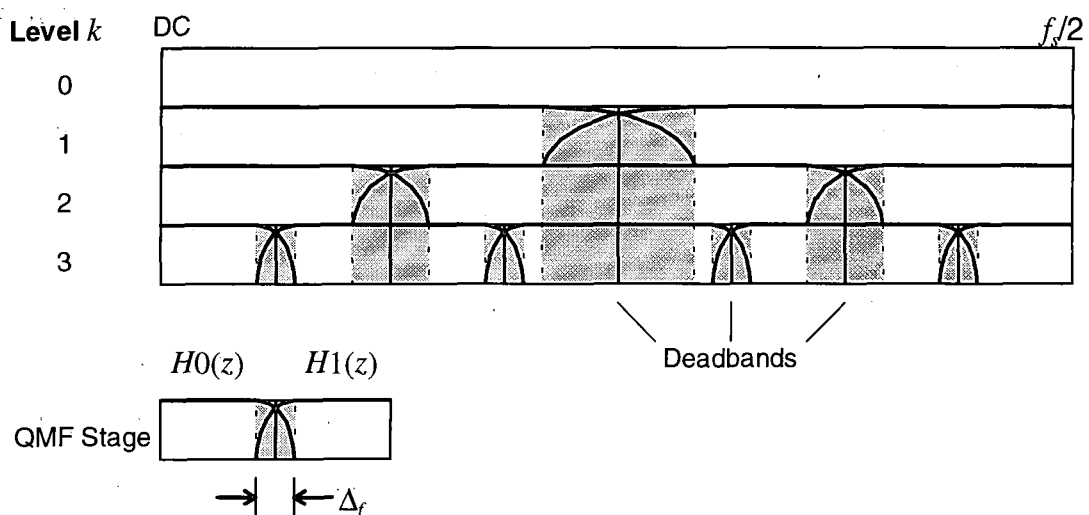
Figure 4.9   Deadbands in Spectral Hierarchy due to Non-infinitesimal Δ_f

The normalisation schemes outlined in this chapter are, as yet, insuffient to produce a functional MAS paradigm. A critical side-effect, as Fig. 4.9 illustrates, is that gradual passband rolloff appears in the transition widths of a practical QMF stage (due to non-

infinitesimal Δ_f) which causes attenuation towards the high frequency end of $H0$ and the low frequency end of $H1$. As discussed in section 3.1.1, an oscillator with time-varying frequency must not enter this region because it will suffer fading as a function of frequency, giving rise to a parasitic amplitude modulation effect that will affect the quality of perceived tone. Also, a sideband will also appear in the transition region, mirrored in $z=\pi/2$, which is boosted above the inaudible level of stop-band gain (δ_s) with the danger of becoming perceptible. Fortunately, this eventuality is prohibited by marking the transition widths as "deadbands", and restricting oscillator allocation to filter passbands which is permissible because time-invariant filterbanks are presumed with constant Δ_f.

The effect of deadbands is not isolated to individual QMF stages; sibling stages inherit the deadbands of their parents, falling under their "shadow" in absolute frequency terms. This is because an oscillator may be correctly interpolated through several passbands before entering the deadband of a stage further up the hierarchy due to cascaded, multistage interpolation. These deadband effects are eliminated by extending the allocation policy of section 3.2.3. Oscillators that are predicted according to their a priori parameters $\{f_{min}(x), f_{max}(x)\}$ to be in danger of falling into a deadband are promoted to the subband that lies immediately above the deadband thus logically excluding transition widths from the MAS paradigm: a formal algorithm is proposed in Chapter 5.

4.7 Review

The PM-FIR and PA-IIR designs outlines offer two efficient alternatives to QMF implementation. The former has poorer Δ_f performance but is linear-phase. The practical problems of achieving functional transparency are identified. Frequency normalisation requires trivial pre-processing of frequency envelope breakpoints after subband allocation. Latency normalisation by delay lines is practical if the QMF is linear-phase, but invalid for the phase non-linear PA-IIR, though envelope pre-skewing is a feasible solution. The most difficult issue is phase normalisation; intractable in a PA-FIR and only valid in a PM-FIR for the series $s_{k,l}$:$\{l=1\}$. Finally, non-ideal filter behaviour manifests

itself as an impinging of a non-infinitesimal Δ_f upon the idealised MAS subband hierarchy which creates deadbands where oscillators cannot be allocated.

5. On the Exclusion of Filterbank Deadbands

5.1 Overview

As observed in section 4.6, the subband hierarchy of a practical QMF filterbank has deadbands which must be excluded from the MAS paradigm in order to guarantee high-quality synthesis. *Logical exclusion* means that deadbands are excluded preemptively by a modification to the MAS allocation policy outlined in section 3.2.3. An alternative is to employ a filterbank in which deadbands are eliminated so that the subband hierarchy is that of the idealised case illustrated in Fig. 3.7. Such a technique may be described as *physical exclusion* and can be achieved by exploiting the properties of complex signal representation in a novel form of oversampled synthesis filterbank, termed the Physical Exclusion Filterbank (PEF) for convenience; the development and evaluation of which the latter part of this chapter is devoted to.

Note that the conclusion of Chapter 3, in that a hierarchical interpretation of a binary-tree subband decomposition is the optimum paradigm for MAS (in terms of both efficiency and flexibility), is accepted *per se* and forms the context for PEF design. There are two principal motives for the PEF concept. The first is to maximise the range of filterbank solutions available to MAS in terms of cost and performance: the PEF forms a useful counterpoint to the QMF approaches. Secondly, as noted in section 4.5, phase normalisation is difficult to achieve for the FIR, and unfeasible for IIR, QMF's. In order to address the desired ideal of complete phase transparency and prove that it is possible in MAS, the PEF is demonstrated to have this unique property. However, the first topic is an analysis of the effect of logical exclusion upon a conventional QMF filterbank.

5.2 Logical Exclusion in QMF Filterbanks

5.2.1 A Formal Allocation Algorithm

Given an oscillator x parameterised by $\{f_{min}(x), f_{max}(x)\}$, eqn. (5.1) is true for subband $s_{k,l}$ in a QMF filterbank paramaterised by Δ_f if $s_{k,l}$ completely bounds x. Using the subband hierarchy notation of section 3.2.2, the first clause pertains to $odd(l)$ which represents

the *H*0 subbands of QMF stages in which the deadband reduces $f_{max}(s_{k,l})$. Conversely, the second clause pertains to even(*l*) which represents the *H*1 subbands where the deadband has the opposite effect of increasing $f_{min}(s_{k,l})$. Note that $a(s_{0,1},x)$ is true by default because it is classical AS. To take into account deadband inheritance, the optimal allocation level *k* in the subband hierarchy is given by the formal condition that for all $i:\{0{\leq}i{\leq}k\}$ there exists a value of $l:\{1{\leq}l{\leq}2^i\}$ for which $a(s_{i,l},x)$ is true and that, conversely, for all $i:\{k{<}i{\leq}K\}$ there exists *no* value of $l:\{1{\leq}l{\leq}2^i\}$ for which $a(s_{i,l},x)$ is true. The optimal subband, given *k*, is therefore designated by the case of $a(s_{k,l},x)$ that is true for a single value of $l:\{1{\leq}l{\leq}2^k\}$.

$$a(s_{k,l},x) = \begin{cases} (k > 0) \wedge \mathrm{odd}(l) \\ \wedge(f_{min}(x) > 2^{-k-1}f_s(l-1)) \wedge \\ (f_{max}(x) < 2^{-k-1}f_s(l-2\Delta_f)) \end{cases} \vee \begin{cases} (k > 0) \wedge \mathrm{even}(l) \wedge \\ (f_{min}(x) > 2^{-k-1}f_s(l-1+2\Delta_f)) \\ \wedge(f_{max}(x) < 2^{-k-1}f_s l) \end{cases} \vee (k = 0)$$

$$(5.1)$$

5.2.2 Allocation Maps for Logical Exclusion

Logical exclusion may be interpreted as an "allocation map" which is a function of *K* and Δ_f with Cartesian co-ordinates $(f_{min}(x), f_{max}(x)/f_{min}(x))$. It is assumed that *K*=3. A point on the map denotes the optimal subband for *x* according to the algorithm of section 5.2.1. The first map, illustrated in Fig. 5.1, is that for an ideal subband hierarchy with Δ_f=0.0 for comparison with the second map which has the realisable figure of Δ_f=0.05. Observe that the allocation patterns illustrated in section 3.2.3 are horizontal "slices" through the Δ_f=0.0 map. The maps also help visualise the gradual transition in the proposed MAS paradigm from supporting fixed-pitch notes in the terminal integer series to those which are low-Q and require large pitch variation as discussed in section 3.2.3.

Note that, in Fig. 5.1, the severest deterioration in the subband hierarchy, as Δ_f increases from Δ_f =0.0→0.05, is in the terminal integer spaced subbands series $s_{k,l}:\{k=3\}$ which are prone to the inheritance of ancestor deadbands, causing a marked increase in the cost of allocating a fixed-pitch note. Indeed, a complete terminal integer spaced subband series

of bandwidth less than $\Delta_f/2$ is not possible as those either side of $f_s/4$ fall completely within the deadband from level $k=0$, which is another limitation on K. In contrast, note, that the relative extra cost of allocating a note with large pitch variation into the fully-overlapping octave spaced subband series $s_{k,l}$: $\{l=1\}$ is slight as each inherits only the deadband of its immediate parent which causes a negligible reduction in passband width of $\times(1-2\Delta_f)$.
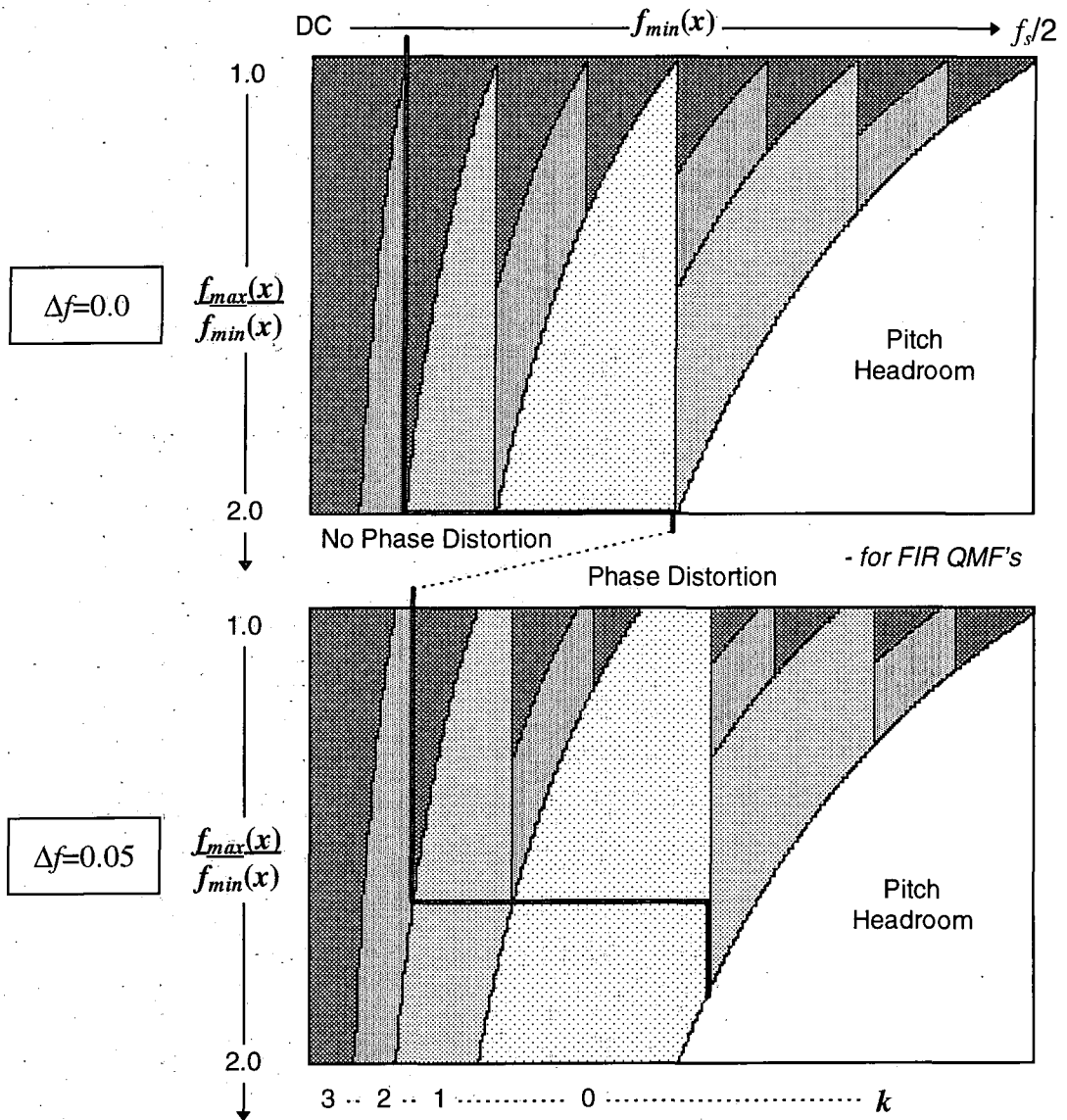
Figure 5.1   *Logical Exclusion Allocation Maps for $\Delta_f=0.0$, $\Delta_f=0.05$ (K=3)*

5.2.3 The Implications of Allocation Maps for PM-FIR and PA-IIR Filterbanks

Fig. 5.1. also differentiates between subbands $s_{k,l}$:$\{l=1\}$ which involve a pure low-pass filterbank path of $H0$'s which has the property (as introduced in section 4.3.3) that baseband phase is equivalent to output phase. For all other subbands $s_{k,l}$:$\{l>1\}$ the filterbank path involves phase-distorting $H1$'s. The frequency threshold between these two subband sets is illustrated to be a function Δ_f and $f_{max}(x)/f_{min}(x)$. This threshold is also improved, interestingly, by wider Δ_f. The point of section 4.3.3. - in that partial phase relationships are most important, and should be preserved, at lower frequencies and that this requirement has a convenient correspondence with PM-FIR QMF filterbank functionality - is reinforced by Fig. 5.1. The allocation map for the PA-IIR filterbank is close to the $\Delta_f=0.0$ map, but with nil phase transparency in all subbands.

5.3 Physical Exclusion Filterbanks

QMF filterbanks are critically sampled. This is a desirable property because it minimises computation. However, the requirement for logical exclusion of deadbands causes a cost increase for oscillator computation because many oscillators are promoted to higher subbands than would be the case with an ideal hierarchy. The fundamental idea of the Physical Exclusion Filterbank (PEF) proposed in this section is that oversampling permits deadbands to be placed outside the signal baseband. In order to achieve this, extra flexibility is required in how signals may be manipulated in frequency domain and this functionality is provided by the properties of complex number representation. As low-pass filters (akin to $H0$ in the QMF) are used, signal basebands rather than sidebands are interpolated leading to the property of complete phase transparency (see section 4.3.3). Unlike the QMF, the PEF has no complementary analysis filterbank as it is an application-specific interpolation structure for MAS. A complex-output multirate oscillator bank is required but there are several efficient algorithms, such as CORDIC, for computing complex sinusoids which are documented in Chapter 7.

5.3.1 Discrete-Time Complex Signals

Fig. 5.2 illustrates the spectrum of a discrete time-complex signal sampled at f_s for which a frequency $e^{j\omega}$ may be conceptualised as a rotating phasor that is anticlockwise for

positive ω and clockwise for negative ω. In conventional "single-phase" sampling, where the quadrature imaginary component is absent and zero by default, a spectrum of bandwidth β (e.g. the sum of oscillators in a subband) resolves into a complex-conjugate pair of sidebands symmetric about integer multiples of f_s and therefore to avoid aliasing, must be sampled at $f_s=2\beta$: the definition of Nyquist's theorem. This relationship is re-interpreted for complex signals where positive and negative frequency are distinct. The minimum sample rate is $f_s=\beta$, but a *couplet* of samples - (*real, imaginary*) - is required for each sample interval and so the data bandwidth is conserved. The baseband span for such a signal is not from DC to β/2, as in the single-phase case, but from -β/2 to +β/2 with DC exactly mid-baseband. These requirements are restatements of Double Sideband (DSB) and Single Sideband (SSB) modulation (Oppenheim and Schafer, 1989).
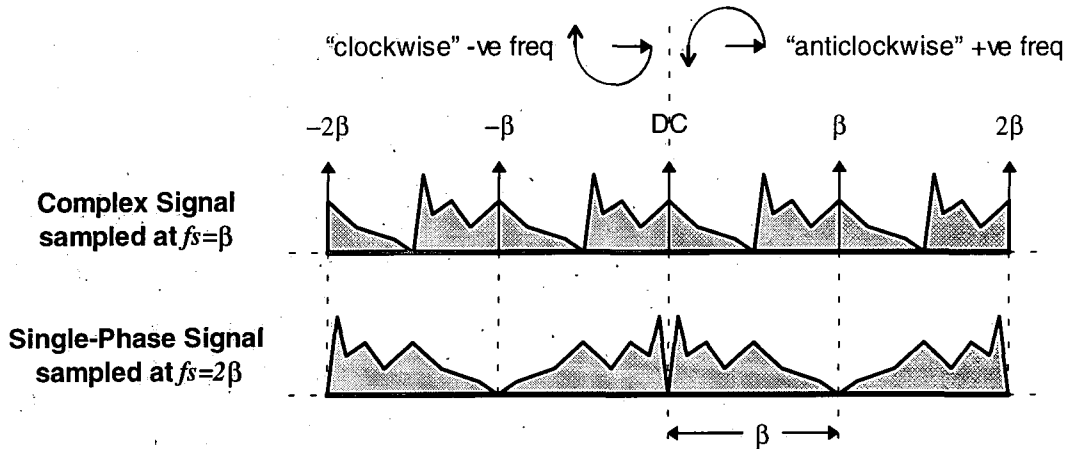
Figure 5.2   Spectra of Discrete-Time Complex and Single-Phase Signals

5.3.2 PEF Filterbank Design

The frequency domain operation of a PEF stage illustrated in Fig. 5.3 is analogous to the QMF synthesis model of section 3.2.1 in that two baseband signals $X0(z^2)$ and $X1(z^2)$, both of bandwidth β, are interpolated by ↑2 (to generate $X0'(z^2)$ and $X1'(z^2)$) and then concatenated in frequency domain to create an output $Y(z)$ of bandwidth 2β. A fundamental characteristic of the PEF is that both $X0(z)$ and $X1(z)$ are oversampled by a factor λ:{λ>1}. After ↑2 interpolation (real and imaginary signal components are filtered in separation by identical filters) a frequency shift is performed of $-\omega_h$ for $X0'(z^2)$, such that the upper frequency bound of the $X0'(z^2)$ baseband coincides with DC. Similarly, a

complementary frequency shift of $+\omega_h$ for $X1'(z^2)$ causes the lower frequency bound of $X1'(z^2)$ to coincide with DC. $\pm\omega_h$, the heterodyne frequencies, are related to λ by $|\omega_h| = \pi/2\lambda$.
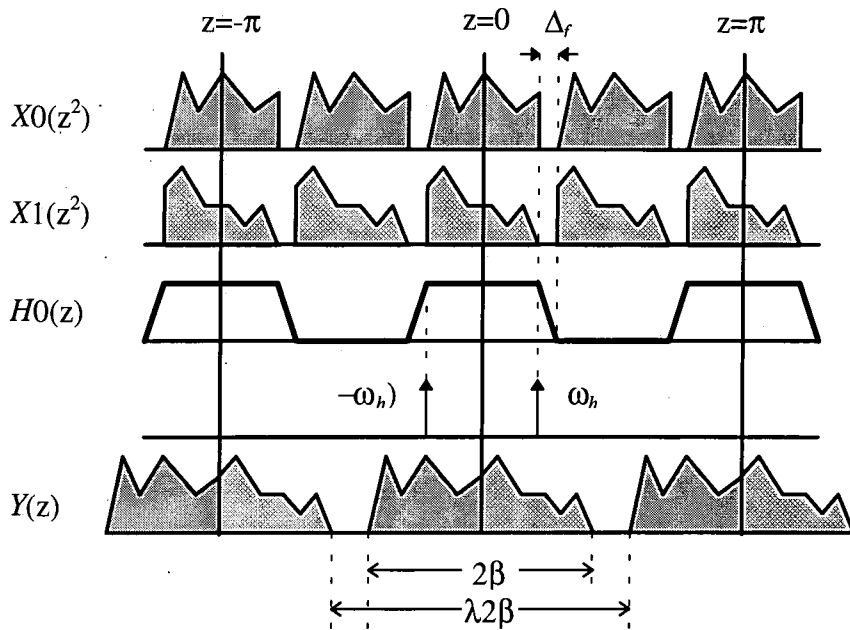
Figure 5.3   Frequency Domain Operation of PEF Stage

Superposition of the interpolated and shifted input spectra creates a perfect concatenation of basebands at DC and therefore eliminates the deadband. The transition width of $H0$ is physically excluded by mapping it on to the guard-bands between input sidebands created by oversampling. Like $X0(z^2)$ and $X1(z^2)$, output $Y(z)$ is a complex signal oversampled by λ and may therefore be input to the $X0(z)$ or $X1(z)$ input of another PEF stage. One of the hallmarks of the PEF approach is that sample rates are $\times\lambda/2$ that of the corresponding QMF stage. If $\lambda<2$ then oscillator bank PWL envelope uncompression is reduced compared to the QMF case, which is significant in that minimisation of this bottleneck is a traditional goal of AS optimisation. The transfer eqn. of the PEF stage is expressed by eqn. (5.2) where '*' denotes frequency-domain convolution.

$$Y(z) = ((X0(z^2)H0(z)) * -\omega_h + ((X1(z^2)H0(z)) * \omega_h \tag{5.2}$$

Fig. 5.4. illustrates the structure of the PEF stage. ↑2 interpolation of the four input signal components is carried out by a FIR filter that is functionally equivalent to the optimised $H0$ of the PM-FIR QMF stage described in sections 4.2.1 and 4.2.2. By a careful choice of λ, ω_h is a rational fraction of π and may be generated by a short LUT. Frequency shifts are carried out by a complex multiplication using the identity $(a+bj)(c+dj) = (ac-bd)+(ad+cb)j$ which constitutes four multiplies per $y[m]$. Therefore, in contrast to a QMF stage which requires i multiplies per $y[m]$, a PEF stage requires $\lambda(2i+4)$ taking into account the PEF/QMF sample rate ratio of $\lambda/2$ and is thus more expensive to compute.
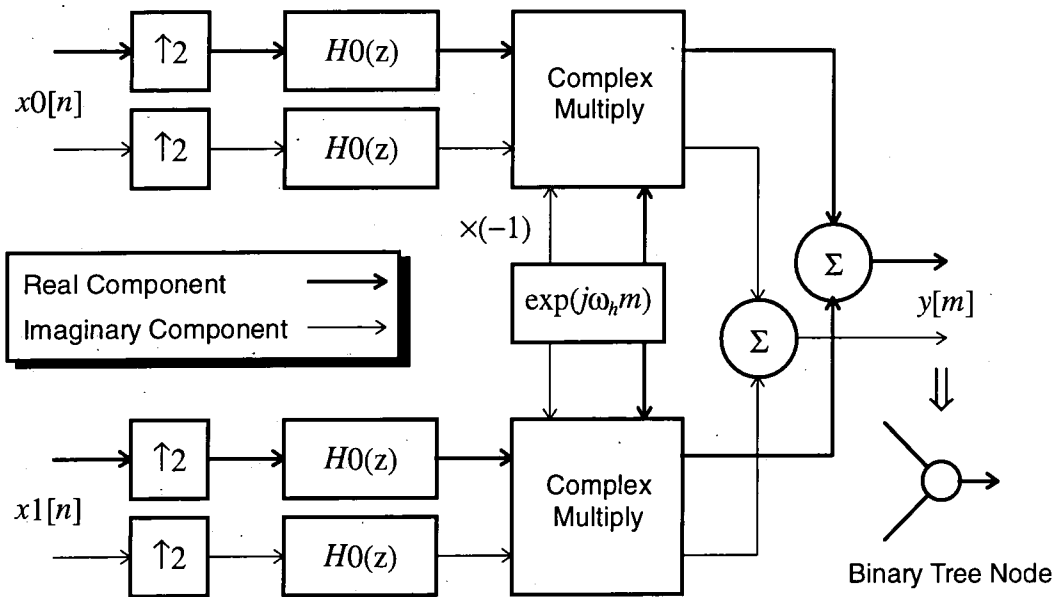
Figure 5.4   PEF Stage Dataflow
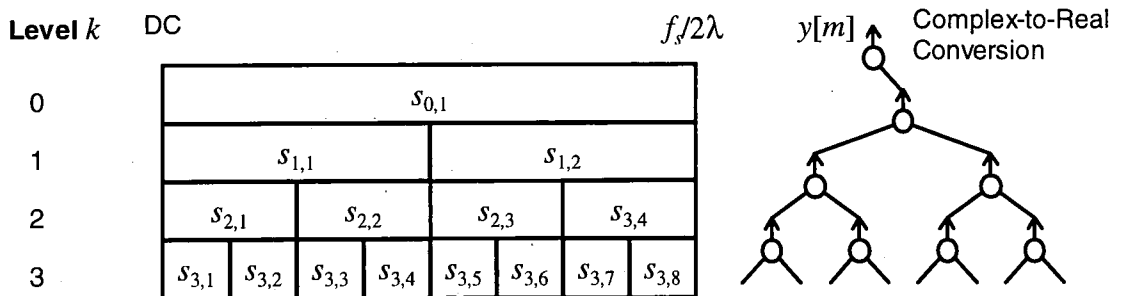
Figure 5.5   PEF Filterbank Topology for Depth K=3

In the filterbank tree of Fig. 5.5 of depth $K=3$, a final complex-to-real stage is necessary in order to generate digital audio. This is achieved by a PEF stage with a single $x1[n]$

input: the complex signal is interpolated and shifted to be positive frequency as before. However, the imaginary component of $y[m]$ is discarded causing a conjugate image of $x1[n]$ to appear in place of the negative-frequency shifted baseband of $x0[n]$: thus $y[m]$ has the desired characteristics of a single-phase signal. There is a direct mapping of ideal subband hierarchy to the corresponding filterbank tree because no frequency-inverting $H1$ filters are used. Also, it can be appreciated how the insertion of a complex-to-real stage "halves" sample rates in comparison to the corresponding subbands in the QMF case of Fig. 3.7, though, in practice, oversampling creates a practical figure of $\times\lambda/2$.

5.3.3 Determination of Oversampling Factor λ

As the PEF is designed to be phase-transparent, linear-phase FIRs must be used in order to avoid the complexity of phase-equalisation in phase non-linear PA-IIRs. λ directly determines the FIR design for $H0$ via substitution of relationship $\Delta_f=(1-1/\lambda)/2$ in eqns. (4.4) to (4.6) resulting in the inverse proportionality of M versus λ illustrated in Fig. 5.6. Clearly, a value of $\lambda\cong1$ results in an FIR with unacceptable latency and computational cost, though oscillator bank redundancy is minimised as it approaches critical sampling. At the other extreme, $\lambda\gg1$ results in an FIR of diminishing latency and cost, but with high redundancy in an oversampled oscillator bank.
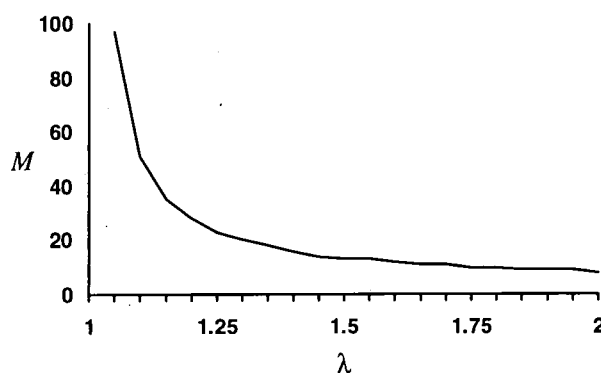
Figure 5.6   FIR M versus λ

Such characteristics are a parallel to generalised MAS economics as summarised in eqn. (3.1). A compromise value is necessary. Experience from simulations (see section 5.3.8) suggest that $\lambda=1.25$ is a good choice for a PEF filterbank with depth $K=3$, requiring an

FIR of M=23 (Δ_f=0.1). f_s= 2λ×20kHz= 50kHz in order to provide a full audio bandwidth of DC-20kHz, though a slightly lower industry-standard rate such as DAT @ f_s=48kHz would be utilised in practice giving an audio spectrum ceiling of 19.2kHz. Another reason for a rational value of λ=1.25 is that ω_h=0.4π and may thus be generated by a trivial five couplet LUT (e^{0j}=$e^{2\pi j}$, 5×0.4π=2π, QED).

5.3.4 Latency Normalisation

The scheme of section 4.3.1. applies equally well to the PEF filterbank, except that two delay-lines in parallel are required for real and imaginary signal components. A PEF filterbank has a latency $2T_{fb}(K)/\lambda$ (×1.6 for λ=1.25) of the corresponding PM-FIR QMF with stages of equal M due to the lower sample rates of the PEF filterbank.

5.3.5 Frequency Normalisation

There are two differences in the frequency normalisation scheme for a PEF filterbank compared to that of section 4.3.2. for a QMF filterbank. The first is that there are no frequency inversions caused by high-pass filtration via $H1$. Secondly, the subband normalised digital frequency envelope $\Omega_x[m]$ of an oscillator x allocated to subband $s_{k,l}$ satisfies -$\omega_h \le \Omega_x[m] \le \omega_h$ and thus may be positive or negative frequency as expressed in eqn. (5.3). After processing through the PEF filterbank, x emerges with the expected single-phase frequency $F_x[m]$.

$$\Omega_x[m] = (1-2a)\omega_h \text{ where } a = \left(\frac{F_x[m] - f_{min}(s_{k,l})}{f_{max}(s_{k,l}) - f_{min}(s_{k,l})} \right) \tag{5.3}$$

5.3.6 Phase Normalisation

Fig. 5.7 illustrates the path of a signal $x[n]$ in a PEF filterbank from an arbitrary subband $s_{k,l}$ to the output $y[m]$. The vector $\mathbf{v}$:{$v_i \in$ {−1,+1}}, where 0≤i≤k, defines the top-down path to $s_{k,l}$ through the binary tree of the filterbank by specifying the heterodyne frequency direction at each stage: v_0=+1 denotes complex-to-real conversion. Latency-normalisation is shown (null for $k=K$) and PM-FIR $H0$ filters for ↑2 interpolation are conceptualised as delay-lines of length M. As there are no $H1$ filters, the phase-distortion

discussed in section 4.3.3. is eliminated. However, a mechanism is required to cancel the time-variance introduced by the heterodyne oscillators. An initial insight to a solution lies in the functional equivalence of the path $\mathbf{v}$ to single-stage interpolation by $I=\uparrow 2^{k+1}$ - with a delay of T samples obtained from eqn. (5.4) - followed by multiplication by a single oscillator ω_{eq} equal to the sum of the constituent heterodyne frequencies in $\mathbf{v}$ as expressed in eqn. (5.5).

$$T = (2^{K+1} - 1)M \qquad (5.4)$$

$$\omega_{eq} = \omega_h \sum_{i=0}^{k} 2^{-i} \mathbf{v}_i \qquad (5.5)$$
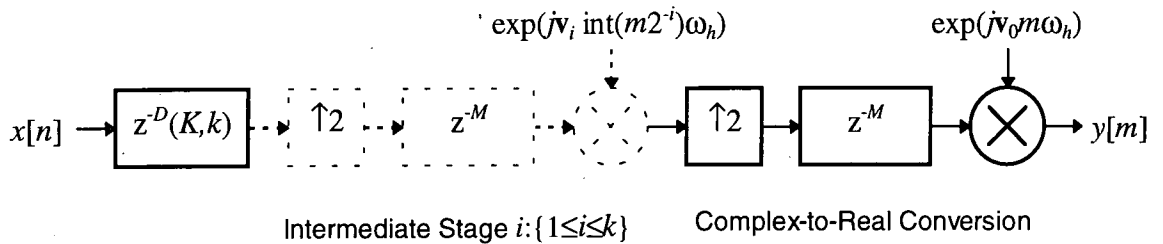
Figure 5.7   Path Model from Level k in a PEF Filterbank

To enable an oscillator x to emerge at $y[m]$ at a specific time $m=t$ with correct phase (given $\mathbf{v}$ and k), the phase offset introduced by heterodyning (denoted $\theta[t]$) is predicted and its negative used to initialise $\Phi_x [t-T]$ and thereafter, the PEF filterbank is phase-transparent upon condition that the partials for a note are accurately phase-accumulated under decimation using eqn. (4.10). $\theta[t]$ is given by eqn.(5.7) with ω_{eq} from eqn. (5.5) and $\theta[T]$ from eqn. (5.6) which originates in the assumption that the instantaneous phase of all heterodyne oscillators at $m=0$ is zero, and by tracing the signal path to the output $y[m]$ at $m=T$ and summing the phase of each heterodyne frequency as it is encountered. A constraint on t is that $t=T+Dc$ where $D=2^{k+1}$ and c is an integer.

$$\theta[T] = \sum_{i=0}^{k} \mathbf{v}_i \, (2^{K-i+1} - 1) M \omega_h \qquad (5.6)$$

$$\theta[t] = (t - T)\omega_{eq} + \theta[T] \qquad (5.7)$$

5.3.7 Functionally Equivalent Parallel-Form PEF Filterbanks

A complete binary-tree PEF is only one of a number of possible topologies and, as hinted in section 5.3.6, it is feasible to construct a parallel-form PEF with an individual interpolator and heterodyne oscillator for each subband. Their outputs are summed and undergo complex-to-real conversion. Consideration of the parallel-form is significant as single step interpolation for a high integer factor $\uparrow I$ has two efficient implementations as discussed in section 3.1.2. Table 5-1 records the performance, determined by a spreadsheet model, of the functionally equivalent multi-stage and single-stage parallel forms of a complete binary tree PEF of depth $K=3$, for input $\lambda=1.25$ and PM-FIRs of $\delta_p=\delta_s=0.0001$ (-80dB) derived using the design eqns. (4.4) to (4.6).

	Single-Stage	Multi-Stage	Binary Tree
Max Latency (ms)	7.86	4.86	6.90
Multiplier B/W (MHz)	6.3	4.5	3.1

Table 5-1   Equivalent Filterbank Performance for K=3

In latency terms, the performance ranking from best to worst is (1) multi-stage, (2) binary-tree and (3) single-stage. The binary-tree requires the lowest multiplier bandwidth because adding an extra level $K \rightarrow K+1$ doubles the number of PEF stages, but since they operate at half the sample rate of the ancestor level the computational cost is proportional to K. In contrast, all interpolators in a parallel form have the same final sample rate. The regularity of a binary-free based upon a single prototype stage implies a simpler control structure than that required by the variety of filters in a parallel form. In conclusion, a binary-tree topology appears the optimal choice for a PEF implementation.

5.3.8 Simulation

In order to validate the theoretical PEF design and its normalisation schema, a simulation was written in 'C' under UNIX. The program included a binary-tree PEF of depth $K=3$, with PM-FIR stages of latency $M=23$ and an allied complex multirate oscillator bank which could be initialised to generate classical AS waveforms. A sawtooth function with fundamental frequency ω_0, as expressed in eqn. (5.8) where t is the time index, is a satisfactory test signal as it includes every harmonic partial, thereby maximising the number of oscillators in the composite waveform, and also making phase distortion phenomena manifest itself readily in the waveform envelope. The imaginary component is taken from the complex-to-real stage of the PEF because eqn. (5.8) is a sum-of-sinusoids. In the simulation, all harmonics up to 20kHz are included.

$$f_{sawtooth}(\omega_0, t) = \sum_{i=1}^{\infty} \frac{\sin(it/\omega_0)}{i} \qquad (5.8)$$

Fig. 5.8 illustrates the output - $y[m]$ *versus* sample index m - for a 110Hz note (A2) assuming f_s=50kHz with frequency normalisation, but excluding that for phase. The characteristic envelope of a sawtooth is corrupted by incorrect phase relationships. Fig. 5.9 includes the phase normalisation scheme of section 5.3.6 and validates the functionality of the proposed technique by generating a high purity sawtooth. Additionally, frequency normalisation as proposed in section 5.3.5 is verified. The latency of 345 samples or 6.9ms @ f_s=50kHz, as obtained via eqn. (5.4), is evident as is the transient response of combined filterbank paths to a step amplitude envelope. As the note is stationary, partials are allocated into the integer-series subbands at level $k=3$: all oscillators are operating at an effective sample rate of $50kHz/2^{k+1}$ =3.125kHz in comparison to the uniform f_s>40kHz of a TOB. Latency normalising delay lines are redundant in this example.

To validate phase accumulation in MAS according to the scheme of section 4.3.2, a 'chirped' sawtooth with linearly increasing frequency is an appropriate signal: the Fourier series is truncated to avoid partials aliasing at the maximum fundamental frequency. Fig. 5.10 illustrates an example of a chirp from 27.5Hz to 440Hz; four octaves from A0 to

A4. The output sawtooth maintains phase integrity under chirping. In contrast, partials in this example are allocated into the fully-overlapping octave spaced subbands series $s_{k,1}$ disposed across layers $k=0..3$. Latency normalising delay lines are therefore employed and demonstrated to operate correctly. The net conclusion of these and many other runs is that the binary-tree PEF permits a MAS paradigm with high functional transparency as desired in the performance criteria of section 1.2.1.

5.4 Review

A 'logical exclusion' algorithm is proposed to overcome the problem of deadbands in a QMF subband hierarchy. A side-effect is that oscillator computation is rendered more expensive in proportion to the dominance that deadbands play in the resulting subband hierarchy: high for the PM-IIR and low for the PA-IIR. In consideration of the phase normalisation problem of the QMF, a novel alternative filterbank design - the PEF- is proposed. Simulation validates that it achieves high functional transparency. However, the PEF has the highest computational overhead of filterbanks envisaged for MAS. The perceptual relevance of phase is the key determinant in which approach is adopted; a subjective issue and of an importance that cannot be dismissed lightly when high-fidelity synthesis is required. It has not been the purpose of this chapter to recommend a prescriptive solution. However, a PM-FIR QMF with limited phase transparency, as discussed in section 4.5, may prove an acceptable compromise between the extremes of PA-IIR QMF and PEF designs.

5.5 PEF Simulation Results
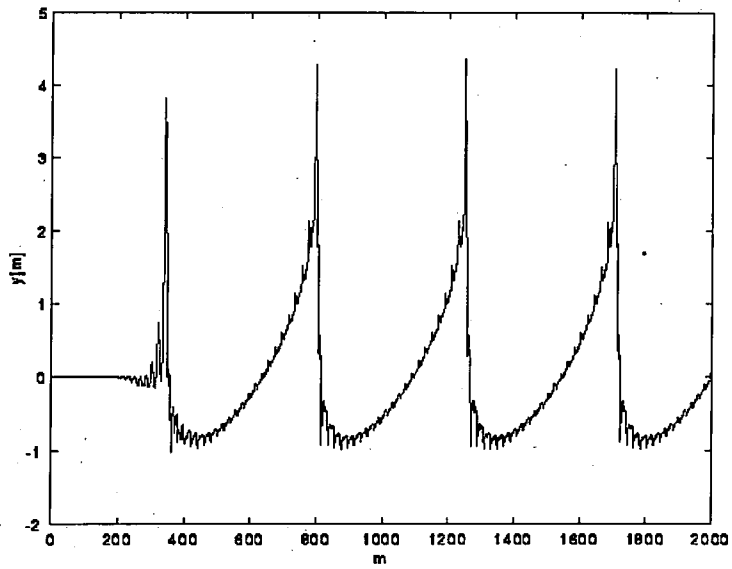
Figure 5.8   110Hz (A1) Sawtooth without Phase Normalisation: y[m] versus m
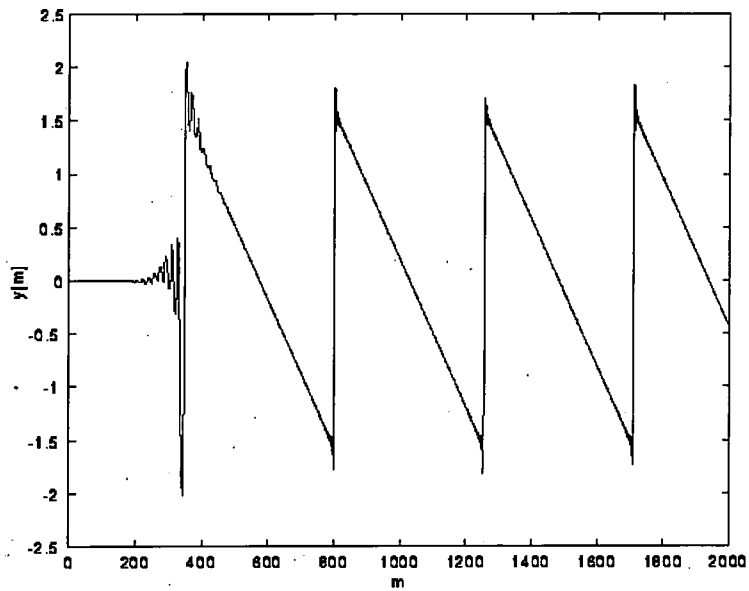
Figure 5.9   110Hz (A1) Sawtooth with Phase Normalisation: y[m] versus m
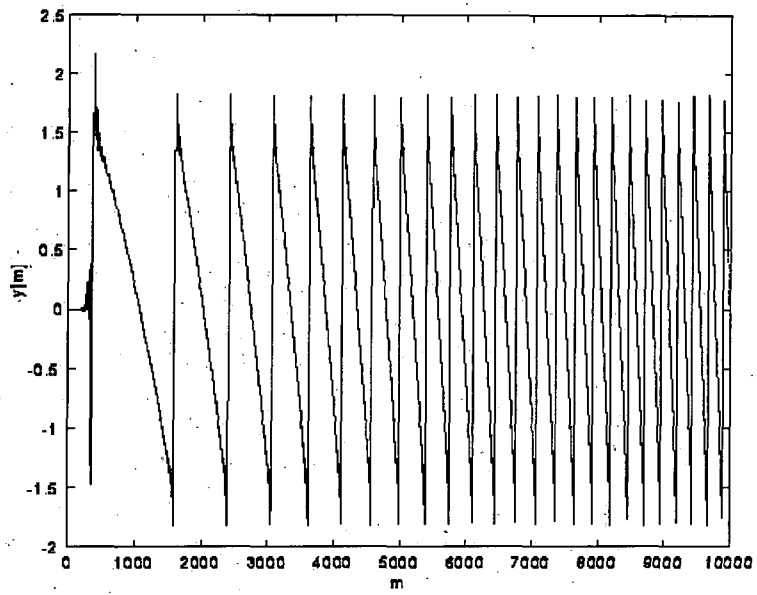
Figure 5.10   27.5Hz to 440Hz (A0 to A4) Chirped Sawtooth: y[m] versus m

6. On Predicting a Performance Benchmark

6.1 Overview

MAS is a statistical optimisation based upon certain assumptions about frequency stationarities in note-based music. It is a challenge to provide a definitive performance benchmark in comparison to classical AS because the principle of generality - a characteristic strength of AS - is retained and cost of computation is adapted to the application in question in an attempt to reach a compromise between the conflicting points of section 1.2. For instance, some specialist sounds (e.g. Sheppard Tones) (Moore, 1990) require mapping into the highest level subband $s_{0,1}$ of the proposed hierarchy with poor performance due to the overheads of supporting an unexploited multirate paradigm. In contrast, a classical 'pipeless' organ (Comerford, 1993) with fixed pitch notes may map with great efficiency into the terminal integer subband series $s_{3,1..8}$ for $K=3$.

In the context of commercially marketed digital music synthesisers, note-based music will remain the pre-eminent application for the foreseeable future whereas parallel fields of music technology research, (e.g. granular synthesis) are oriented towards facilitating the composition of 'avant-garde' electro-acoustic music. The research motive of MAS is in concordance with the former imperative, and hence pieces of music in traditional score notation provide the most appropriate statistical source for benchmarking. Score notation is based upon discrete pitches with occasional expression marks for pitch inflection such as vibrato and glissandi; such accentuation is sparingly used in practice, or restricted to small sections of ensembles. Hence this investigation is confined to fixed-pitch note synthesis, though a limited account is taken of average pitch dynamics.

Eqn. (3.1) in section 3.1.7 forms the basis for quantifying the relative efficiency of MAS relative to classical AS. However, three items of data are required for its evaluation, two of which are u_{mas} and $v(K)$. The third item n_k is application-specific and for benchmarking, the best choice is 'representative' partial frequency distributions derived from actual musical scores; in particular, those composed for large ensembles such as

isolated exceptions such as the muted trumpet). However, this region does have major significance in the perception of transient phenomena and percussion (Masri and Bateman, 1996).

The database structure of SHARC has a simple two-level hierarchy. At the top level are 39 directory names (listed in Appendix A) formed of abbreviations of each instrument and playing style (analysed in separation, hence four entries for the cello). Within each sub-directory is a 'CONTENTS' file which has a row for each analysed note and columns listing comprehensive parameters from each analysis. Each row contains a filename which references an ASCII file in the same subdirectory which describes the spectral envelope of the timbre by a list, starting from the fundamental, of partial magnitudes (in dB's) and phases. Magnitude is relative to the highest energy partial which is normalised to 0dB. The format of the CONTENTS file is standardised and facilitates automated database searching, which is exploited in the next section.

6.3 Allocation Simulation for Orchestral Synthesis

6.3.1 Overview of Experiment

The first stage of benchmarking is to derive an estimate of expected allocation statistics from orchestral music in the form of a histogram. Fig. 6.1 illustrates how such an allocation simulation divides into three sections; (1) SHARC pre-processing via FBIN.C, (2) representation of orchestration and score, and (3) the simulation program ALLOCSIM.C itself. These are described, respectively, in sections 6.3.2 to 6.3.4. Results are presented in section 6.3.5. The entire experiment was coded in 'C' under UNIX. The histogram quantifies the mean number of oscillators required per time unit during the piece of music *versus* an ascending series of frequency bins spanning the audio spectrum, numbering n_{bins}. An estimate is thus provided of the computation required for AS as a function of frequency band. Fine frequency resolution is desirable so that the histogram may be accurately superimposed upon a QMF subband hierarchy which includes logically-excluded deadbands. Therefore n_{bins}=256 was chosen as a compromise between computational complexity (as data is analysed in a spreadsheet) and simulation

accuracy. The upper frequency limit of the histogram is determined by SHARC at 11.025kHz.
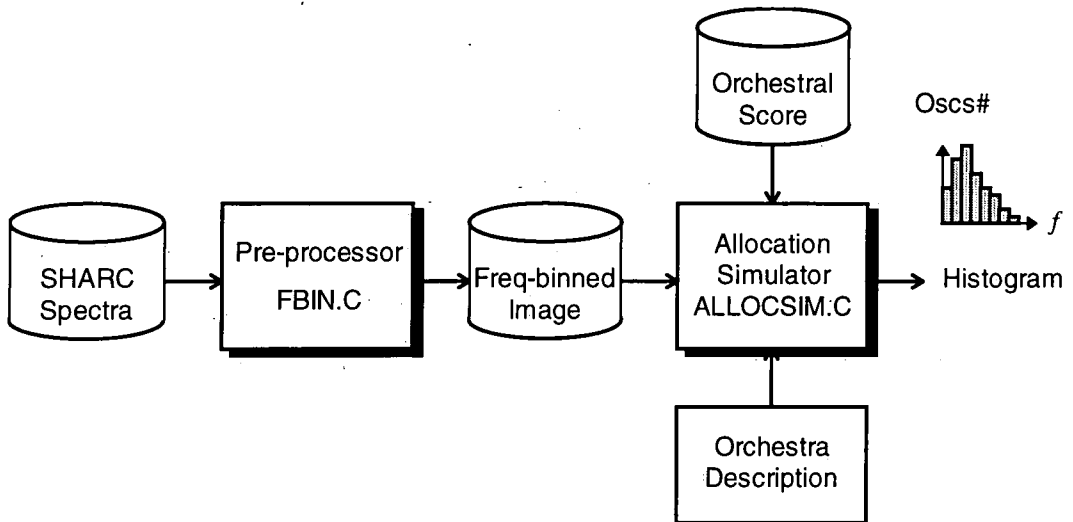
Figure 6.1   Dataflow of Allocation Simulation

6.3.2 Preprocessing of SHARC Data

The data record format of SHARC is not suitable for immediate input into ALLOCSIM.C because spectra are organised on the basis of an equal-tempered musical scale. Therefore a pre-processor FBIN.C was written to translate SHARC into 'image' with an integer-series of n_{bins} bins that matches the format of the required histogram. The image has the same fundamental directory structure as the source database, but with the difference that each note file comprises a list of n_{bins} values denoting the number of oscillators required per bin. A problem is that an arbitrary note has a theoretically infinite partial series and to include all those below the Nyquist limit of $f_s/2$ would generate an equal number of oscillators per bin: computation would be dependent on note pitch rather than timbre rendering the exploitation of SHARC redundant. Some method is required, based upon a perceptual criterion, for optimising the number of partials actually synthesised to the timbre in question.

In traditional AS, the cost of computation grows linearly with the number of partials as summarised in eqn (1.2). However, many will have insufficient power to add to the

subjective tone quality constituting redundant computation. An optimal partial set is generated by sorting partials in terms of perceptual significance and sequentially including them into the set until an acceptable level of synthesis quality is reached. This requires imposition of an auditory model which is a non-trivial task in the context of ensemble synthesis (c.f. section 2.3). A simpler 'pruning' technique was adopted in FBIN.C. Partials are sorted in terms of amplitude and included into the set until the power ratio of included to excluded partials exceeds a quality threshold ε, which may be interpreted as an S/N measure with the excluded residual signal as 'noise'. Given a series of partial amplitudes A_i:{$1 \leq i \leq N$}, the RMS power P of the resulting time-domain signal is given by eqn. (6.1) which is used as the pruning criterion.

$$P = \frac{\sqrt{\sum_{i=1}^{N} A_i^2}}{\sqrt{2}} \qquad (6.1)$$

The resulting image of each SHARC timbre is a function of spectral envelope, note pitch and also ε. As ε is a variable in the simulation, a number of SHARC image databases are compiled over a pertinent range of ε in order to observe how MAS allocation efficiency is affected as a function of synthesis quality. In the simulation a range of ε∈ {40,60,80}dB was chosen to illustrate this point. As ε→∞, few partials are excluded and therefore the impact of timbre information in SHARC has little effect on the histogram as explained previously. Also, there will be redundancy in the computation of partials which, individually, contribute relatively little to the overall quality of the perceived timbre: for example, those with low energy partials in the top octave from 10kHz to 20kHz. However at ε ≅ 0dB, few partials are included and synthesis quality will be unacceptably low. These observations imply that there is an optimal value of ε that provides the best histogram for benchmarking. In conclusion, it is pertinent to emphasise that the application of eqn. (6.1) is empirical, but that the experimental results (Figs. 6.3 and 6.4) corroborate with our intuition in that increasing the synthesis quality (ε) causes an upward sweep in the amount of AS resources dedicated to the region above 5kHz which can be interpreted as increasing 'sparkle'.

6.3.3 Organisation of Score and Orchestration Data

The source chosen for score data were oversize 'full' scores from the Music Section of the University of Durham Library. These list the orchestral parts vertically and are read in a single pass across each page, representing the bar-by-bar activity of parts in parallel. This data requires translation into a machine readable format for use by ALLOCSIM.C. MIDI files are a readily available resource but are limited by the functionality of target hardware (e.g. PC sound card) and the range available over the INTERNET was considered unsatisfactory. For instance, MIDI has only 16 channels and ALLOCSIM.C uses a minimum of 26 parts. Manual score entry gave greater control over the origin and complexity of the music input into the allocation simulator.

To this end, a graphical editor program was written in MS QBASIC on a PC that permitted an intuitive, fast and visually verifiable means of entering score data bar-by-bar. The output 'part' file is ASCII and consists of a list of triplets describing {*note start time in beats from start of piece, MIDI pitch, note duration in beats*}. Polyphony is represented by permitting notes on consecutive lines to have the same start time; a feature which is required, for instance, when a violin section which is usually monophonic becomes biphonic. Though the rate of score entry was initially slow, after refinements to the editor and accustomisation to the routine, progress accelerated vindicating the idea.

Sir Edward Elgar's Variations on an Original Theme: Enigma (Op. 36) was chosen as the source score for a number of reasons (Elgar, 1899). Each variation is relatively short and thus the burden of score entry was lightened: a full symphony is a task of questionable utility. Also, there is a wide range of orchestration density between variations, permitting a tentative analysis of the relationship of histogram to style. Therefore two were chosen which use full orchestral resources at some point and represent contrasting extremes of expression; (1) No. 1 'C.A.E.' - 'light', and (2) No. 9 'Nimrod' - 'intense'. Appendix B tabulates the parts required, to which SHARC timbre they relate, and the number of voices chosen to make up each part. The Late Romantic orchestration is typical for the date of composition (Westrup and Harrison, 1959). A match is found for all parts except the timpani.

An important point is raised here point about how MAS is used to build up ensemble textures. For instance, SHARC includes an analysis of a violin ensemble. In reality, individual violins in a section are never precisely in tune and hence the ensemble sound cannot be truly described by a single harmonic partial series. It is the interaction of many voices in imperfect consonance distributed over an acoustic space that generates the characteristic rich texture (Meyer, 1993). MAS facilitates polyphony in the example violin section by reducing the cost of synthesising many instantiations of a single prototype. Also, the control problem cited in section 1.1 is reduced by exploiting hierarchy in the application structure. Instead of n independent instantiations, n variants of the prototype are initialised and related to the latter by intuitive metaparameters e.g. a detuning control (Jaffe, 1995).

To recreate the spatial distribution of individual voices is more difficult because the computational efficiency of MAS (explained in section 3.1.3) relies on the fact that the number of voices exceeds the number of filterbanks, requiring that voices are grouped and post-processed as a single composite signal. Even if independent voice synthesis were feasible, the overhead of supporting reverberant post-processing for each voice is extravagant. In the context of an orchestra, the instruments of a section are located closely together and the variation in impulse response between players to a distant listener will not vary significantly. Group post-processing of the section is therefore likely to be of acceptable fidelity, though sub-division of a large section with a set of filterbanks is an optional refinement.

6.3.4 ALLOCSIM.C Algorithm

Initialise histogram array $h[1..end\ of\ piece, 1..n_{bins}]$ to zeros

For each part p

Read in SHARC image $b[min..max\ register, 1..nbins]$ relating to p

Set of currently sounding notes $N=\{\}$

For each time instant $t=1..end\ of\ piece$

Initialise new notes in N from part file of p starting at t

Remove any notes in N terminating at t

For each $n \in N$

$$h[t, 1..n_{bins}] = h[t, 1..n_{bins}] + (b[n, 1..n_{bins}]*(\text{voices in } p))$$

Compute required statistics from h

ALLOCSIM.C is based upon a simple sequencing algorithm which is summarised in the above pseudocode. It takes as arguments ε (to address the specified SHARC image) and a directory containing the part files of the specified piece. At each iteration of the main loop ALLOCSIM reads a new part file (determined by the orchestration of Appendix B) and models the set of notes sounding at each time instant (quantised to semiquavers for the Enigma Variations). A histogram associated with each instant is incremented with the predicted frequency bin allocation of oscillators for that part so that histogram evolution throughout the piece is recorded. This enables the post-computation of other statistics than a mean histogram, if so desired. A problem encountered in execution was that some timbres are missing from SHARC resulting in an incomplete compass for some of the parts. This was resolved by locating the nearest note at frequency f_n to the desired note at f_d which has a SHARC timbre and copying its image. To take into account the pitch difference, each bin of the desired note is multiplied by f_n / f_d.

6.3.5 Analysis of the Allocation Histograms

The results of the simulation for the two pieces are plotted in Figs. 6.3 and 6.4 in section 6.6 in three-dimensions to express the fact that the number of oscillators required is a function of both frequency and ε. Though n_{bins} is set to 256 in the simulation, the resulting histograms are 'noisy' and are therefore smoothed in the graphs by using 16 major bins, each comprising the sum of 16 minor bins. This may be compared with the frequency quantisation of a subband hierarchy of depth $K=3$ which has 4 subbands spanning DC-10kHz, which is too coarse for illustrating the trends involved.

The first observation is that though the two pieces are different in character, they display a marked degree of correlation between their respective histograms. The only significant difference is that the allocation density in 'Nimrod' is higher than in 'C.A.E.' as expected because the former has greater orchestration density than the latter. In the limit at

ε=80dB, 'Nimrod' requires ×1.48 the resources of 'C.A.E.'. The correlation is attributable to the common orchestration, the pruning artefacts of SHARC, and to a much lesser extent, by the shared provenance of the pieces.

At low values of ε (e.g. 20dB), where only a few high amplitude partials are included in the synthesis set, the related histogram displays a steep roll-off showing that synthesis resources are concentrated in the lower frequencies below major bin 8 (approximately 5kHz). Above this threshold, negligible resources are required. As ε is increased towards 80dB, the rolloff becomes progressively shallower until it is almost flat. This is because more low amplitude high-frequency partials are included into the set, constituting increasing 'detail' in the treble region above 5kHz and therefore a higher quality of synthesis, thus confirming the observations of section 6.3.2.

At this point, it is appropriate to discuss two caveats which impinge upon the results subsequently presented. The first is that only stationary spectra are considered and that the synthesis overheads of transients, which are nonstationary and perceptually important during note attack, are absent from the histograms. Essentially, transients are short noise bursts with a flattish spectrum implying increased energy at higher frequencies relative to the 'low-pass' nature of SHARC spectra: the requirement for substantial AS resources above 5kHz is indicated. This could be realised by setting ε→∞. However, eqn. (6.1) is based upon assumptions of stationarity. One solution (c.f. section 3.3) is the generation of broad-band energy through AM or FM (Jansen, 1991), (Fitz and Haken, 1995). Hypothetically, the AS resources required are relatively low as only a few nonstationary oscillators, selectively placed, are needed to span the whole audio spectrum in contrast to the scenario where ε→∞.

The second problem concerns the validity of the pruning algorithm of eqn. (6.1): the intuitive assumption that a partial's energy relates to its perceptual significance is complicated by masking effects. However, the validity of imposing an auditory model upon an individual timbre is relevant only when the note is heard in isolation. For orchestral synthesis, it should be imposed upon the complete oscillator set for the orchestra and this philosophy is eschewed for the reasons set out in section 6.1. In these circumstances therefore, the chosen algorithm, despite its empiricism, appears quite

justifiable and is supported by the evidence of histogram behaviour as a function of ε in Figs. 6.3 and 6.4. It should be emphasised that the experimental methodology of this chapter represents a 'first-cut' investigation into AS resource allocation, for which there is a lack of published work and yet a good potential for substantive research, and it is to be hoped that more advanced analytical methods could be developed.

6.4 Benchmark Calculation

6.4.1 Quantifying MAS Overheads

A problem with eqn. (3.2) for predicting a benchmark is that values must be attributed to u_{mas}, u_{as} and $v(K)$ in the absence of a full MAS implementation. How are reliable values to be estimated and according to what standard are they to be measured? The commonest measure for quantifying the cost of numerical algorithms is the number of multiplies required for a given task. For a linear signal processing system such as a filterbank, the frequency of multiply / accumulate (MAC) operations required is a highly suitable measure. However, the overheads of oscillator bank computation are dominated by the computation of $\sin(\Phi[n])$ - a non-linear calculation - and in managing the control data bandwidth of PWL envelope uncompression. A single MAC alone is required per output sample for imposition of $A_i[n]$ and for the accumulation of S sinusoids in eqn. (1.1).

The cost of transforming an oscillator bank from an AS to MAS paradigm is minimised by the binary-tree subband decomposition of the filterbank proposals of Chapters 4 and 5 which have a small set of $K+1$ sample rates related by powers-of-two integer ratios. As explained in Chapter 9, the imposition of MAS requires multirate scheduling to support multiple oscillator sample rates, but the complexity increase over that which is required for a TOB is shown to be of not overwhelming significance: a latency penalty is the most important side-effect. As proposed in section 1.4.3, MAS seeks to preserve the optimality inherent in the TOB model and therefore it is considered reasonable to assume that $u_{mas} \cong u_{as}$. The principal computational overhead of MAS is represented by filterbank computation.

$$E1 = \frac{\text{net number of AS oscillator updates per second}}{\text{net number of MAS oscillator updates per second}} \qquad (6.2)$$

$$E2 = \frac{\text{net number of MAC's in AS per second}}{\text{net number of MAC's in MAS per second}} \qquad (6.3)$$

To avoid the eventuality of incommensurate cost measures leading to an unreliable value of E, it was decided to decompose eqn. (3.1) into two benchmarks, $E1$ and $E2$, summarised in eqns. (6.2) and (6.3). $E1$ measures the performance increase caused by the net reduction of oscillator sample rates in MAS compared to AS based on the equality $u_{mas} \cong u_{as}$, but excludes filterbank computation. In contrast, $E2$ measures the performance increase caused by the net reduction in the number of MAC's required per second by MAS compared to AS: the denominator includes those required by filterbanks and a MAS oscillator bank. The justification for abstract benchmarks arises from the fact that the real cost measure, as discussed in section 3.1.7, is implementation-dependent (e.g. CPU time or VLSI area): $E1$ and $E2$ provide useful figures which will be a dominant in such a measure and can be evaluated without implementation knowledge. $E1$ is chosen deliberately to illustrate a strong optimisation facet of MAS whereas $E2$ places a "devil's advocate" emphasis on the overheads of filterbank computation in MAS in terms of a standard measure of DSP complexity.

6.4.2 Filterbanks: Quantity, Topology and Cost

Before $E2$ can be evaluated, values must be attributed to the number of filterbanks n_{fb}, and $v(K)$. Appendix B describes a hypothetical allocation of the orchestral instruments required by Enigma Variations to a total of $n_{fb}=8$ filterbanks. Families of instruments (e.g. woodwind) and sections of identical instruments are allocated to individual filterbanks. This illustrates how MAS exploits hierarchy in the synthesis application by factoring the large number of outputs from S individual oscillators in AS into a small number of logically related streams. In the case of an orchestra, an intuitive grouping is by locality in an acoustic volume. A separate 'effects' process is envisaged in Chapter 8 which locates each filterbank stream within a virtual acoustic space in relation to the listener who requires a multichannel signal for perception of the synthetic music.

Level k   DC $f_s/2$

0

1

2

3

Subbands
>11kHz
not required
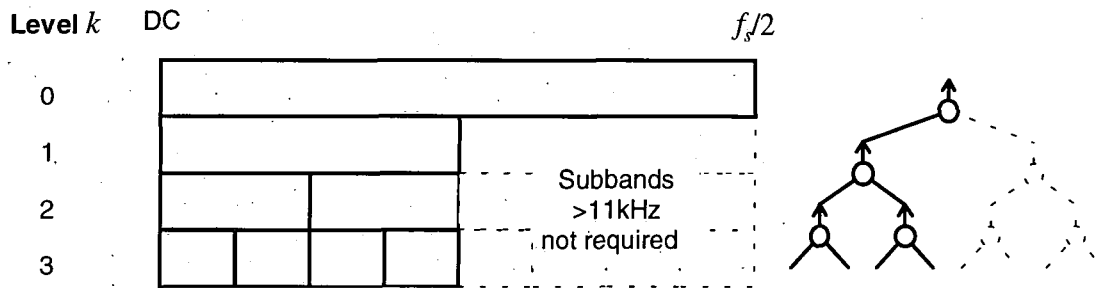
Figure 6.2   Subband Hierarchy and Filterbank Topology Used in Simulation

Each filterbank is assumed to be an incomplete binary tree of the form of Fig. 6.2. This is because SHARC has no information above 11kHz, or $f_s/4$ for a QMF filterbank executing at a nominal CD sample rate of 44.1kHz, and therefore interpolation resources for this upper octave are redundant. Note that it is still feasible to allocate partials with frequencies up to the Nyquist limit of $f_s/2$ to the computationally most expensive subband $s_{0,1}$ for e.g. the synthesis of broadband transients. Such a scenario illustrates the utility of configuring filterbank topology to the partial frequency distribution, rather than imposing complete binary trees. In the PEF filterbank simulation, partials above 10kHz are allocated to $s_{0,1}$ because the upper bound of the subband hierarchy is normalised to 20kHz (see section 5.3.2).

Benchmarking is complicated by the assumption that fixed pitch notes are employed. If this were strictly the case, then all oscillators could be allocated into the terminal integer-series $s_{3,1..8}$ of the proposed subband hierarchy. Players in a real orchestra introduce - as a matter of course - note pitch dynamics, to add expression and texture to the music. Vibrato is the commonest phenomena. However, to quantify the amount required by each instrument in the simulation is highly subjective. A compromise is to introduce a pitch modulation constant of $\pm\tau$ semitones that is applied to all instruments of the orchestra that reflects their mean requirement for pitch dynamics, as documented theoretically in section 3.2.3. Instruments in a real orchestra will span a range of pitch stationarities and τ is thus a rather coarse modelling parameter. Nevertheless, τ allows us to investigate $E1$ and $E2$ as a function of subband allocation pattern.

$$v = v_s \sum_{k=0}^{K} 2^{-k} t_k$$

(6.4)

Filterbank cost v is evaluated in eqn. (6.4) where v_s is the number of MAC's required per output sample in a prototype stage and t_k:{$0 \le k \le K$} represents the number of filterbank stages required to generate the subband set at each level k in the specified topology. The 2^{-k} term represents the relative decimation between filterbank output and level k. For Fig. 6.2, $t_{0..K}=(0,1,1,2)$ for a QMF and $t_{0..K}=(0.5,0.5,1,2)$ for a PEF. $t_0=0$ for a QMF signifies that $s_{0,1}$ is the output fullband and no interpolation is required and $t_0=0.5$ for a PEF signifies that complex-to-real conversion is necessary which is constituted by half of a PEF stage (in practice, as only a real output is needed, the final $+\omega_h$ frequency shift resolves to two MAC's rather than four). Also, $t_1=1$ for a QMF signifies that a complete stage is necessary to generate $s_{1,1}$ ($s_{1,2}$ is available by default but unused) and $t_1=0.5$ for a PEF signifies that a half stage can be used because interpolation of the $H0$ and $H1$ subbands is independent (see section 5.3.2).

Therefore, all three of the filterbanks discussed in Chapters 3 and 4 are benchmarked to evaluate their relative performance. $v_s=4c+8$ for the PEF stage of section 3.2 where c is the cost of a constituent FIR: using the suggested PM-FIR stage design of section 5.3.3. and then the optimal FIR structure of section 3.2.1, $c=6$ and therefore $v_s=32$. For a PM-FIR QMF stage, v_s is calculated via eqns. (4.4) to (4.6) and is a function Δ_f: a range of Δ_f must therefore be benchmarked to determine an optimum value where filterbank computation is balanced with the effect of logical deadband exclusion. For the PA-IIR stage discussed in section 4.4, $v_s=6.5$ with a negligible $\Delta_f \cong 0.01$. From these values of v_s, Table 6-1 records v as determined via eqn. (6.4) with $\delta_s=\delta_p=-80$dB for the PM-FIR.

Filterbank	v
PM-FIR QMF	$\cong 1.152/\Delta_f$
PA-IIR QMF	11
PEF	38

Table 6-1   Filterbank Cost v

6.4.3 Applying Allocation Simulation Data

Interpreting the histograms of Figs. 6.3 and 6.4 reveals that ε=20dB and ε=80dB represent, respectively, poor quality synthesis and an asymptotic level of fidelity with computational overkill. The best compromise lies between, but given that the 'optimal' value of ε will remain obscure without the evidence of listening tests, the histograms of ε=40dB and ε=60dB are selected to provide realistic upper and lower bounds to the region where the optimal histogram is likely to be found. A convenient framework for processing the histogram data is as a spreadsheet as integration of all other simulation parameters, such as logical deadband exclusion, is facilitated.

A histogram is represented as a row vector $\mathbf{h}$ comprising n_{bins} elements spanning DC to 11.025kHz. This is mapped on to an allocation pattern $\mathbf{a}$ (using the specified filterbank topology $\mathbf{t}$ of Fig. 6.2) which is also a row vector of size n_{bins} with identical frequency span denoting which level $k:\{0 \leq k \leq K\}$ each bin of $\mathbf{h}$ is mapped to in the subband hierarchy. Each bin of $\mathbf{a}$ has, rather, the weighting 2^{-k} denoting the decimation caused by allocation to level k in comparison to classical AS in subband $s_{0,1}$. For a PM-FIR QMF implementation $\mathbf{a}$ is a function of Δ_f, which determines the impact of logical deadband exclusion. τ determines the Q of a partial series and whether $\mathbf{a}$ tends towards the terminal integer-spaced series $s_{3,1..8}$ (high τ and Q) or the fully-overlapping octave-spaced series $s_{0..K,1}$ (low τ and Q as illustrated theoretically in section 3.2.3).

$$E1 = \frac{f_{as} \sum_{i=1}^{n_{bins}} \mathbf{h}_i}{f_{mas} \mathbf{a}(n_{bins}, \mathbf{t}, \Delta f, \tau, f_h) \mathbf{h}^T} \tag{6.5}$$

$$E2 = \frac{f_{as} \sum_{i=1}^{n_{bins}} \mathbf{h}_i}{f_{mas}(n_{fb} v + \mathbf{a}(n_{bins}, \mathbf{t}, \Delta f, \tau, f_h) \mathbf{h}^T)} \tag{6.6}$$

$E1$ is expressed in eqn. (6.5) where f_h denotes the upper frequency limit of $\mathbf{a}$ (f_h =22.05kHz for a QMF, f_h =20kHz for a PEF) and is used to normalise bin widths in $\mathbf{a}$ to

those of **h**. Account must be taken of output sample rates. f_{as} relates to classical AS at f_{as}=44.1kHz whereas f_{mas} relates to MAS and is set at f_{mas}=44.1kHz for critically-sampled QMF filterbanks and f_{mas}=50kHz for the oversampled PEF (see section 5.3.3) and thus the cost increase of oversampling in a PEF is incorporated. The assumption that $u_{mas} \cong u_{as}$ holds for complex signals in a PEF because, though oscillator sample rate is halved relative to single-phase representation, the cost of an update is approximately doubled (see Chapter 7). $E2$ as expressed in eqn. (6.6) is based on eqn. (6.5), but includes filterbank costs in the denominator where n_{fb}=8 from section 6.4.2 and v is dependent on the specific filterbank from Table 6-1.

6.4.4 Analysis of Benchmarking Results

The benchmark results are plotted in section 6.6. Figs. 6.5 and 6.6 display the $E1$ benchmarks for 'C.A.E.' and 'Nimrod' respectively, versus filterbank type and τ (the pitch variation tolerance) for both the ε=40dB and ε=60dB histograms: 'PA' refers to the PA-IIR QMF, and PM-FIR QMF Δ_f is shown for four representative values $\Delta_f \in \{0.05, 0.1, 0.15, 0.2\}$. Both graphs are approximately identical because (i) their respective histograms are highly correlated and (ii) filterbank computation is excluded. The PA-IIR gives the best results as the filterbank is critically sampled and provides a near-ideal subband hierarchy. The PEF is a close second because it has an ideal subband hierarchy but is oversampled. For the PM-FIR, it can be seen how $E1$ falls off rapidly with increasing Δ_f due to logical deadband exclusion. The deleterious influence of deadbands increases towards the middle of the hierarchy (on the frequency axis). This why $E1$ for the PM-FIR is lower for the ε=60dB histogram (which has a higher density of partials in this region) than that for ε=40dB, particularly at high values of Δ_f. The effect of τ is to reduce $E1$: the falloff is quite steep for filterbanks close to the ideal subband hierarchy, but shallow for the PM-FIR with large Δ_f because logical deadband exclusion already promotes many partials into higher subbands and thus the extra proportion promoted by large τ is small.

Figs. 6.7 and 6.8 display the $E2$ benchmarks in the same format as those for $E1$. The inclusion of filterbank computation into $E2$ illustrates the economy of scale exploited by MAS in that $E2$ for 'Nimrod' is consistently higher than 'C.A.E.' which has less

orchestration density. The PA-IIR has pre-eminent performance due to its low-cost and near-ideal subband hierarchy, but also has the steepest rolloff of $E2$ against τ for the reasons previously discussed. The PEF has worst performance due to the expense of oversampling and complex-signal representation. However, $E2$ is greater than unity for the PEF, even at high values of τ, indicating that net savings in MAC computation are made. The PM-FIR has a performance intermediate between the PA-IIR and PEF and displays the interesting characteristic of a global maximum of $E2$ versus Δ_f. An explanation is that at low values of Δ_f, filterbank cost (and latency) are excessively high, though a near-ideal subband hierarchy is obtained. Conversely, at high values of Δf, filterbank cost is low but the subband hierarchy suffers severe deterioration due to logical deadband exclusion. The value of Δ_f for which $E2$ is a maximum increases with τ, for the reason that a quasi-ideal subband hierarchy is not required for high τ where most partials are promoted anyway. A value in the region $0.1 \leq \Delta_f \leq 0.15$ appears to bound $\max(E2)$. This factor, in combination with the observations of section 5.2.2 are strong evidence for the optimality of $K=3$.

6.5 Review

The theory of MAS economics advanced in section 3.1.7 is supported. If taken *cum grano salis*, then the benchmarks predicted in this chapter may be expected to reflect those gained by a prototype MAS implementation and extensive listening tests, which is outside the scope of this thesis. $E1$ is an 'optimistic' measure concerned solely with throughput optimisation in a TOB. $E2$ is a 'pessimistic' measure which is weighted towards filterbank computation and includes only the single output MAC operation of a TOB. However, both display consistent values significantly greater than unity supporting the superior optimality of MAS over AS. Performance increase is largely determined by filterbank choice and can be ranked in ascending order as (1) PEF, (2) PM-FIR and (3) PA-IIR which, in descending order, reflects functional transparency as determined in Chapters 3 and 4, thus illustrating the tradeoffs inherent to MAS.

6.6 Benchmark Results
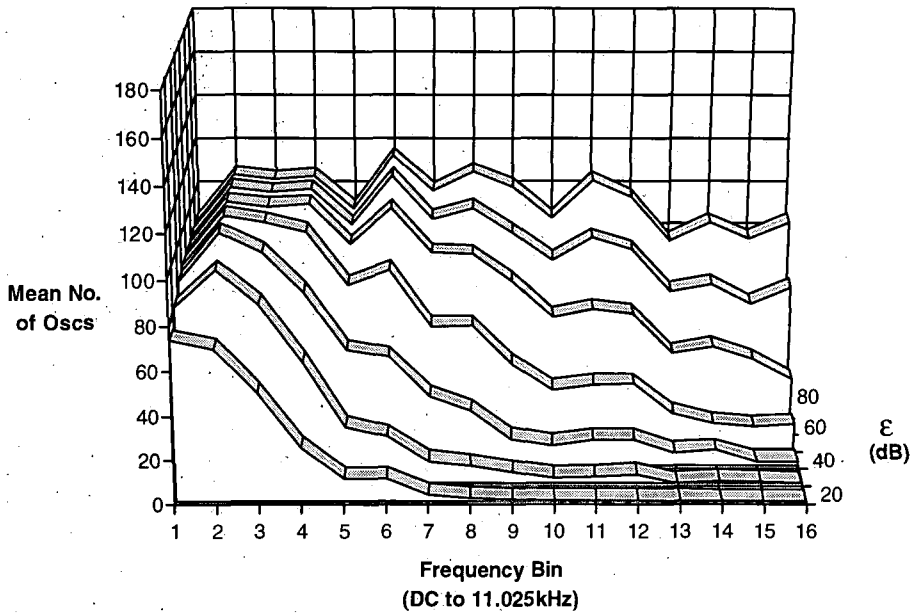
Figure 6.3   'C.A.E.' Allocation Histograms
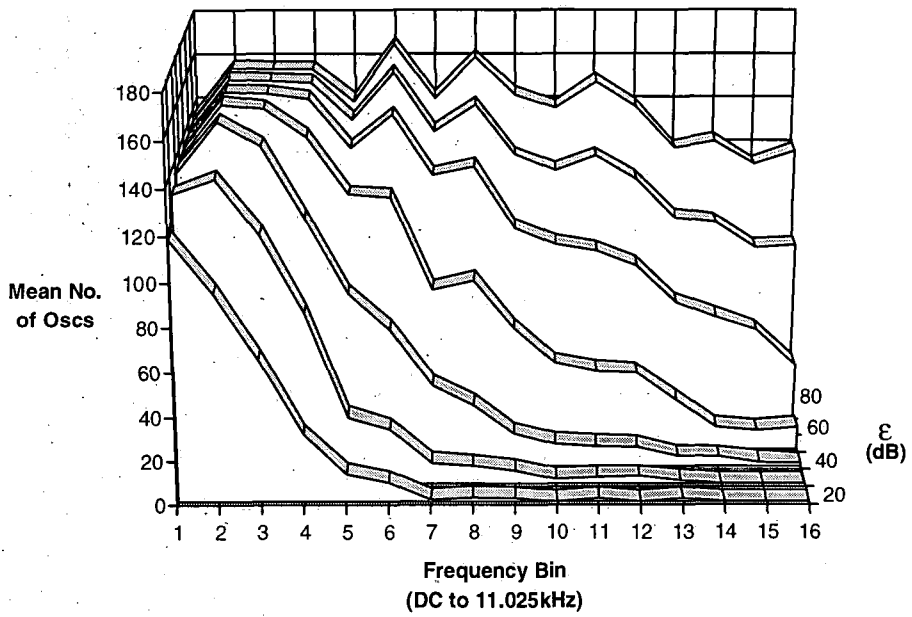
Figure 6.4   'Nimrod' Allocation Histograms
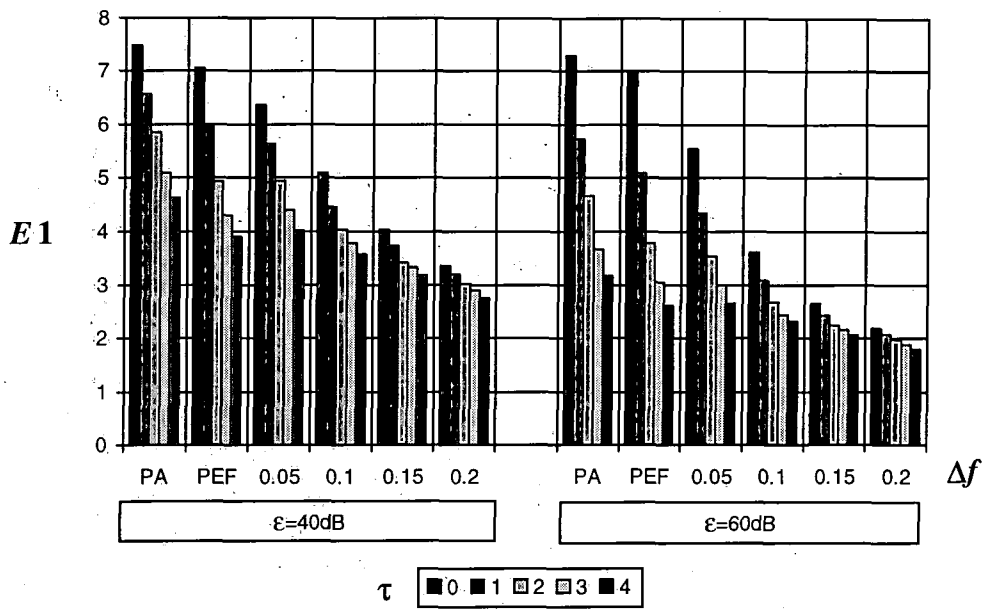
Figure 6.5   E1 for 'C.A.E.'
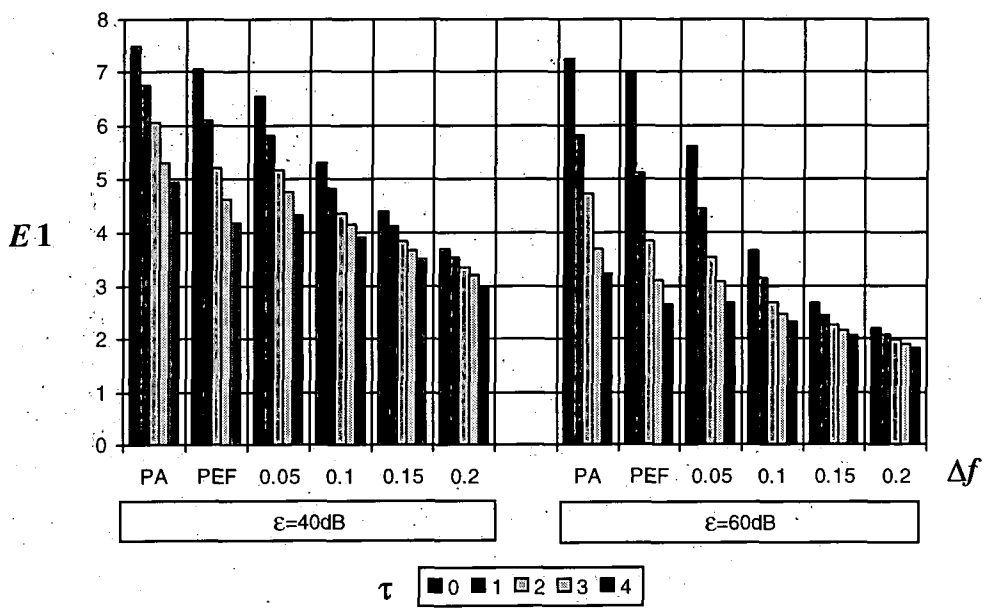
Figure 6.6   E1 for 'Nimrod'
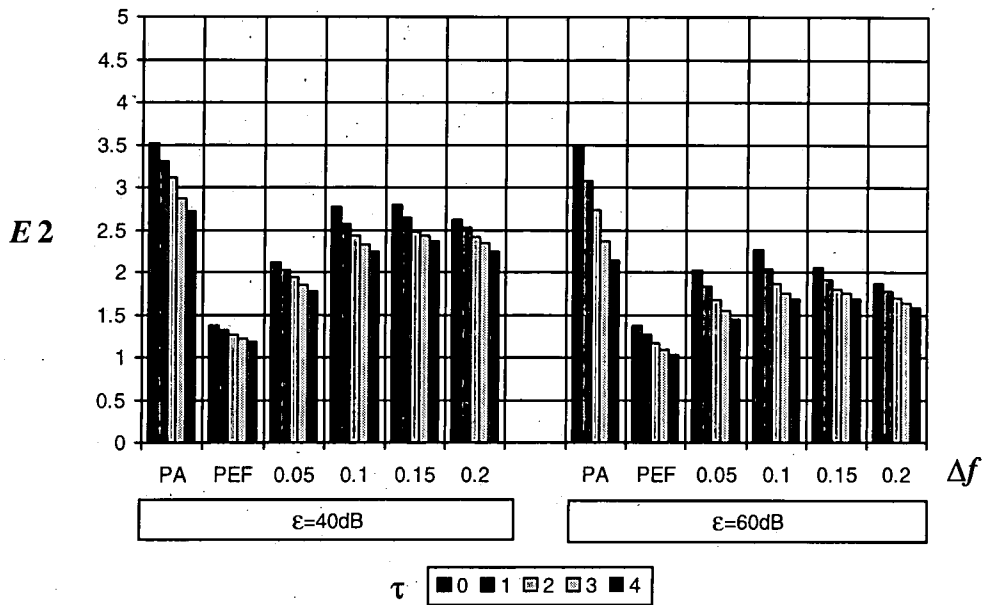
Figure 6.7   E2 for 'C.A.E.'
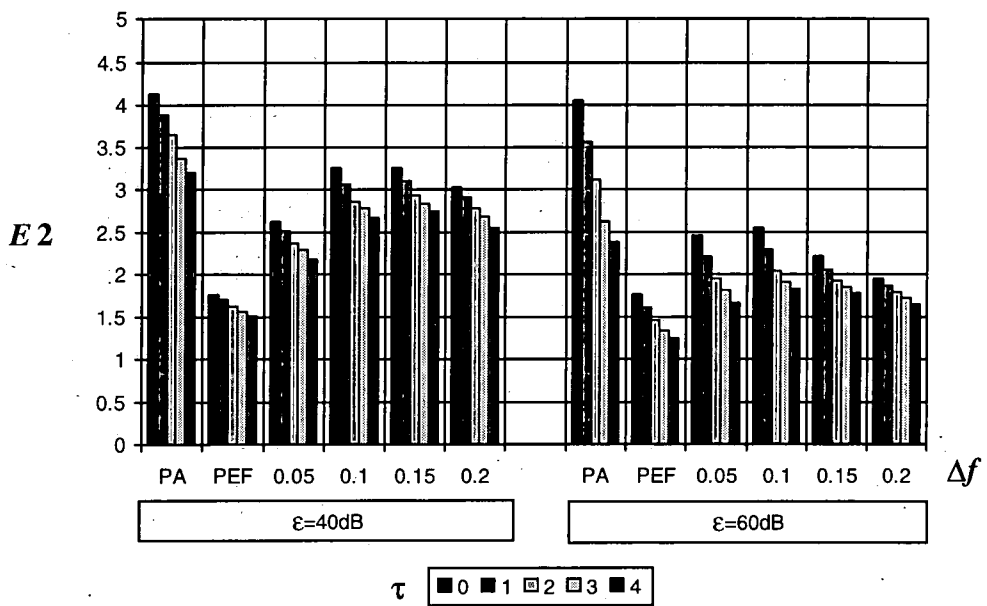
Figure 6.8   E2 for 'Nimrod'

7. Digital Sine Oscillator Design

7.1 Overview

Having determined a MAS paradigm with a variety of filterbank implementations, this chapter is concerned with identifying efficient digital algorithms for the generation of discrete-time sinusoids, with a bias towards their suitability for VLSI implementation, to serve as the central core for the MAS Coprocessor (MASC) documented in Chapters 8 and 9. The topic of sinusoid generation is extensively researched and therefore the early part of this chapter is a literature review. Both single-phase and complex algorithms (for input into the PEF of section 5.3) are considered; the latter is an unusual application in a digital audio context. For this reason, the latter part of the chapter (section 7.5) is devoted to an analysis of a novel application of the CORDIC algorithm for AS / MAS. Nonetheless, there are six desirable properties by which the different algorithms can be assessed (the i subscript of $F_i[n]$ etc. is omitted as a single prototype oscillator only is under consideration):

- Low VLSI area.

- High throughput capacity to minimise the critical bottleneck of the TOB.

- A control rate for $F[n]$ and $A[n]$ at f_s using PWL envelopes.

- Fine-resolution, linear response and wide dynamic range of $F[n]$ and $A[n]$.

- Non-interaction of $A[n]$ and $F[n]$

- An output with a consistent S/N ratio independent of dynamic range.

There are two contrasting schools of sinusoidal oscillator design; (i) recursive and (ii) phase-accumulator oscillators (Higgins, 1990). In general, recursive and phase-accumulator approaches are associated respectively with software and hardware implementations. Recursive oscillators, documented in section 7.1, are discrete-time simulations of physical systems which have simple harmonic motion as their solution. Cost of computation is dominated by the number of multiplies required per sample.

Implementation in DSP's is efficient because of the concentration of effort expended by manufacturers in minimising multiplier delay . A disadvantage for AS is the difficulty of dynamic frequency control at f_s which is necessary for PWL enveloping. In contrast, phase-accumulation as documented in section 7.3 presumes a control rate of f_s. Indeed, it is synonymous with the classical definition of AS in eqn. (1.1). The phase of the sinusoid is computed explicitly from a discrete integration of $F[n]$. The main overhead is accurate sine computation, historically obtained from LUT's.

The choice depends on the context. An important design criterion is the 'S/N' ratio of RMS signal and noise amplitudes: the difference between the approximate and idealised output of a digital sine oscillator represents noise added to the signal which affects the perceived purity of tone. The size of a direct LUT exponentiates with the required S/N and hence if a high-fidelity stationary sinusoid is required, recursive designs are often chosen. Also, as LUT size increases, so does propagation delay and a bottleneck forms in the TOB as discussed in section 1.3.2. However, linear interpolation or CORDIC can significantly accelerate throughput. Another strategy to reconcile the efficiency of recursive oscillators with the controllability of phase-accumulation is to increase the complexity of a recursive system such that it remains stable when the centre frequency is updated at a rate of f_s.

7.2 Recursive Oscillators

7.2.1 The Biquad Oscillator

The simplest recursive system of interest is the second order *biquad* oscillator of Fig. 7.1 (i.e. a quadratic numerator *and* denominator in the z-transform) with coefficients determined by eqns. (7.1) and (7.2) which has a single-phase output $y[n]$ and a z-plane representation of a complex-conjugate pair of poles at $re^{\Omega j}$ and $re^{-\Omega j}$ where Ω is the desired output digital frequency (Higgins, 1990). Only a single multiply is required for constant amplitude when $r=1$ and therefore it is efficient to compute. The amplitude constant and initial phase and are set by seeding $y[n-1]$ and $y[n-2]$ with a past history dependent on Ω. The poles lie exactly on the unit circle because there are no roundoff

errors for subtraction ($b=-1$) and thus the oscillator has excellent long term stability. The set of available frequencies is linearly spaced and quantised by wordlength.

$$a = 2r\cos\Omega$$
$$b = -r^2$$

$$(7.1, 7.2)$$
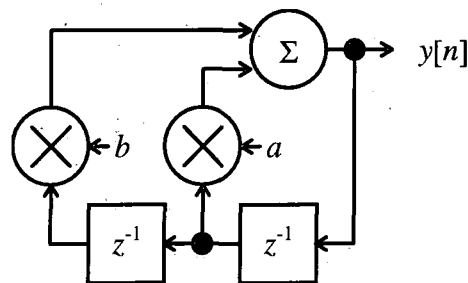
Figure 7.1   The Biquad Oscillator

A disadvantage for AS is that frequency and amplitude cannot be controlled dynamically by changing the coefficients. Due to the second-order nature of the system, the past state $y[n-2]$ must be re-computed. In effect, a new segment of oscillation must be spliced into the previous and so sample-rate resolution control is impractical. The oscillator is constant amplitude and therefore imposition of amplitude requires an extra multiply on $y[n]$. However, a second multiply enables $b\neq-1$ and provides exponential growth ($r>1$) or decay ($0<r<1$) which can be used for simulating primitive envelopes e.g. plucked strings. The pole radius r represents inter-sample gain.

7.2.2 The Coupled Form Oscillator

The coupled form oscillator (Higgins, 1990) illustrated in Fig. 7.2 is a matrix multiplication of a two-dimensional vector expressed in eqns. (7.3) to (7.5) and has a z-plane representation of a single pole at $re^{\Omega j}$. The matrix constitutes both a rotation by Ω and a scaling r of the vector each sample period, with $r=1$ for constant amplitude. Due to quantisation errors, maintaining $r=s^2+s^2=1$ for all values of Ω is not possible and leads to long-term drift towards zero amplitude or overflow: a solution suggested by Higgins (1990) is to detect when $y[n]\cong0$ and reset $x[n]$ to the correct vector length. Abuelhaija and Alibrahim (1986) and Fliege and Wintermantel (1992) discuss more sophisticated

Page 108

methods. Unlike the biquad oscillator, coefficients can be changed at the sample-rate without the necessity to re-compute past states because the system is first order.

$$c = r\cos\Omega$$
$$s = r\sin\Omega$$

<div align="right">(7.3, 7.4)</div>

$$\begin{bmatrix} x[n+1]) \\ y[n+1]) \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} x[n] \\ y[n] \end{bmatrix}$$

<div align="right">(7.5)</div>
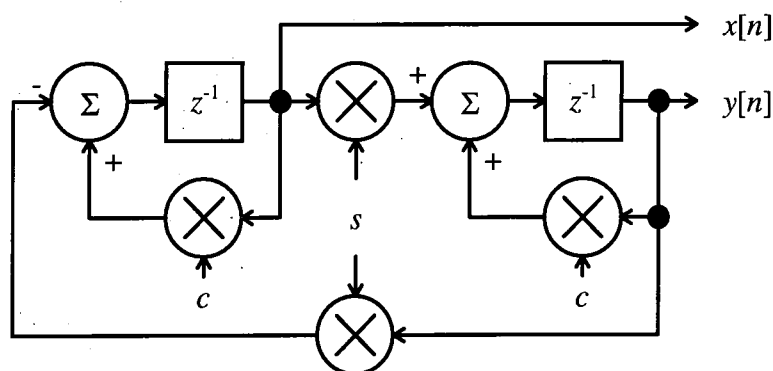
Figure 7.2   The Coupled form Oscillator

The oscillator has a complex output but requires four multiplies per sample compared to one for the biquad. However, it may execute at a sample rate equal to its output bandwidth as explained in section 5.3.1; half that of a single-phase system like the biquad. The output frequency range is therefore $-\pi < \Omega < \pi$. Imposition of $A[n]$ requires an additional two multiplies on the output. An alternative is modulation of r (via s and c) at a rate lower than f_s for a piecewise exponential envelope relying on precision calculation for accurate prediction of oscillator growth / decay over indefinite periods. However, as discussed in section 1.1.3, PWL approximation is pre-eminent for modelling arbitrary envelopes and its exponential alternative has minimal patronage.

For linear control of Ω via $F[n]$, eqns. (7.3) and (7.4) must be computed every sample period because of the non-linear relationship of s and c to $F[n]$. An alternative is to use direct PWL approximations of the envelopes for s and c from $F[n]$ such that eqns. (7.3) and (7.4) require computation only at breakpoints. However, the interdependency of s

and c in that $c^2+s^2=r$ means that r will deviate from unity during PWL spans because of the non-linearity, causing undesirable amplitude modulation. Also two envelopes are used in place of one causing an increase in computation. Imposition of $F[n]$ is thus problematic for the coupled-form oscillator.

7.2.3 The Modified Coupled form Oscillator

A method for solving the inherent instability, difficulty of frequency control and cost of computation of the coupled form is proposed by Gordon and Smith (1995). It relies upon ordering the update sequence such that $y[n+1]$ is dependent $x[n+1]$. Oscillator dataflow is illustrated in Fig. 7.3. Only two multiplies are required per output sample compared to four for the coupled form. The matrix form in eqns (7.6) and (7.7) involves a single variable ε (see eqns. 7.6 and 7.7) that controls frequency which may be changed dynamically at the sample rate. This is because the system is first-order, like the coupled form, and there are no coefficient interdependencies resulting from a requirement to maintain $r=1$ as the matrix determinant (r) is unity by default and independent of wordlength. It is reported by the authors that the facility of dynamic frequency control leads to a primary application to FM synthesis.

$$.\varepsilon = \Omega / \pi \qquad (7.6)$$

$$.\begin{bmatrix} x[n+1] \\ y[n+1] \end{bmatrix} = \begin{bmatrix} 1 & -\varepsilon \\ \varepsilon & 1-\varepsilon^2 \end{bmatrix} \begin{bmatrix} x[n] \\ y[n] \end{bmatrix} \qquad (7.7)$$
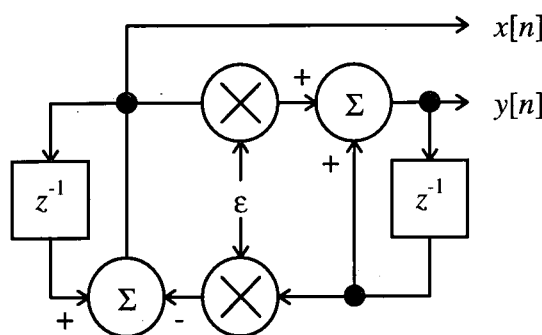
Figure 7.3   The Modified Coupled form Oscillator

A strength of the modified coupled form for AS is that ε is a linear function of Ω (via eqn. 7.6) in contrast to properties of the unmodified coupled form. An analysis of eqn. (7.7) by the authors results in a prediction of $(x[n], y[n])$ given by eqns. (7.8) to (7.10) which depends upon the initial condition of $(x[0], y[0])$. Setting $(x[0], y[0])=(1, \cos(\varphi))$ results in $(x[n], y[n])=(\cos(n\Omega), \cos(n\Omega-\varphi))$ and conversely $(x[0], y[0])=(1, -\sin(\varphi))$ results in $(x[n], y[n])=(\sin(n\Omega), \sin(n\Omega-\varphi))$. When $\Omega\cong0$, $y[n]$ is in approximate quadrature with $x[n]$ since $\varphi\cong\pi/2$ according to eqn. (7.8). However $\Omega\cong\pm\pi$ results in $\varphi\cong0$ and consequently, $x[n]$ and $y[n]$ are in approximate phase. Therefore the modified coupled form is unsuitable for complex sinusoid generation which requires frequency-independent quadrature between $x[n]$ and $y[n]$.

$$\varphi = (\pi - \Omega)/2 \qquad (7.8)$$

$$\mathbf{G} = \frac{1}{\sin(\varphi)}\begin{bmatrix} \sin(n\Omega+\varphi) & -\sin(n\Omega) \\ \sin(n\Omega) & -\sin(n\Omega-\varphi) \end{bmatrix} \qquad (7.9)$$

$$\begin{bmatrix} x[n] \\ y[n] \end{bmatrix} = \mathbf{G}^n \begin{bmatrix} x[0] \\ y[0] \end{bmatrix} \qquad (7.10)$$

7.2.4 The Waveguide Oscillator

A recursive design proposed by Smith and Cook (1992) is the second-order digital waveguide oscillator illustrated in Fig. 7.4. Like the biquad, only one multiply is needed to generate a constant amplitude sinusoid with long term stability. From a control point of view, it is superior as frequency can be changed dynamically without needing to recompute past states. The cost is that an extra compensating multiply, $G[n]$ as derived from eqns. (7.11) to (7.13) must be introduced to normalize amplitude when a change occurs in Ω occurs (i.e. $C[n] <> C[n-1]$) which can be omitted for constant Ω when $G[n]$ is unity ($G[n]$ can be included each iteration, if desired, in order to provide exponential growth / decay determined by $r[n]$).

$$C[n] = \cos\Omega \tag{7.11}$$

$$g[n] = \sqrt{(1 - C[n])/(1 + C[n])} \tag{7.12}$$

$$G[n] = r[n]g[n] / g[n-1] \tag{7.13}$$
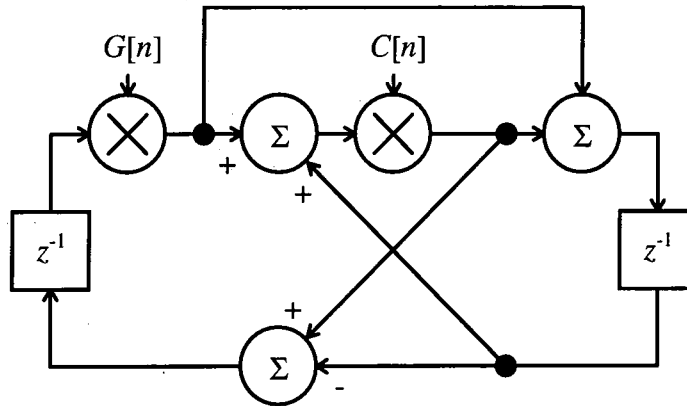
Figure 7.4   *The Second Order Digital Waveguide Oscillator*

The authors assert that such a scheme give the oscillator a controllability which is suitable for FM synthesis at half the number of multiplies required for the modified coupled form. It is also reported that AM as a by-product of FM when $G[n]$ is unity has acceptable side-effects upon timbre. Alternatively, a first order approximation of $G[n]$ is sufficient for amplitude normalisation. For the requirements of MAS, controllability is not a significant improvement on the biquad oscillator. Like the modified coupled form, the design is oriented towards an economic recursive implementation of FM synthesis.

7.2.5 Review of Recursive Oscillator Designs

Recursive oscillators have inherently poor controllabilty that improves with the complexity of the design as summarised in Table 7.1. The coupled form is most expensive and the only one suitable for use with a PEF filterbank. It suffers from stability problems and non-linear frequency control, but as it has a complex quadrature output it may execute at half the sample rate of single-phase oscillators. The biquad and waveguide oscillators are cheapest to compute but have non-trivial frequency control

problems at f_s. The modified coupled-form therefore appears the best choice as it has a single linear frequency control parameter and long-term stability. In all instances imposition of the amplitude envelope $A[n]$ from eqn (1.1) requires extra multiplies on the output. All of the designs use two state variables as is to be expected of any second-order system.

Design	Multiplies per $y[n]$		$F[n]$ control	Complex O/P
Biquad	2	inc $A[n]$	Recompute history	No
Coupled Form (CF)	6	ditto	Non-linear, unstable	Yes
Modified CF	3	ditto	Linear, stable	Not strictly
Waveguide	2	ditto	Non-linear, stable	No

Table 7-1   Summary of Recursive Oscillator Features

7.3 Phase-Accumulator Oscillators
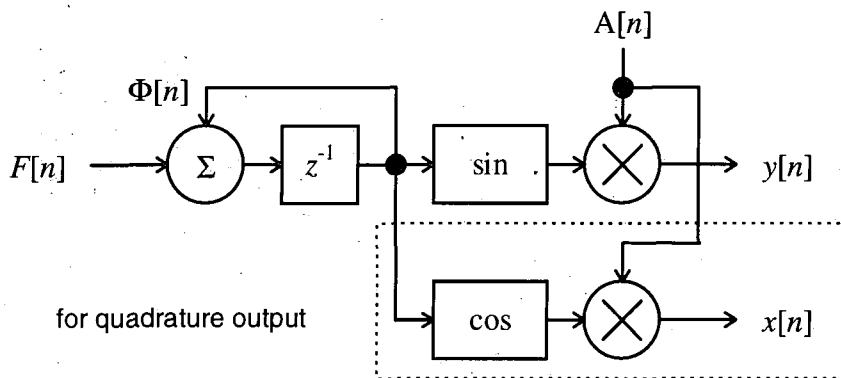
Figure 7.5   The Phase-Accumulator Oscillator

Fig. 7.5 illustrates the dataflow of both the single-phase and complex form of the phase-accumulator oscillator which represents the critical section of the TOB (discussed previously in section 1.3.2). The cost of computing of the complex form (two sine calculations, two multiplies at operating bandwidth) is the same as the single-phase form (one sine calculation, one multiply at twice operating bandwidth). From another

perspective, the former is more economic because the accumulation rate of $\Phi[n]$ is halved as it is a part of the control domain where the rates of $F[n]$ and $A[n]$ are also halved. For PWL envelopes, the number of interpolated envelope samples is therefore halved but does not lead to a reduction in breakpoint bandwidth.

Two properties of phase accumulation are that (i) radians are an inconvenient unit for binary arithmetic, and (ii) indefinite phase accumulation is unnecessary because $\sin(\Phi[n])$ is periodic. A classical solution (Moore, 1977) is to represent $\Phi[n]$ as a free-running two's complement accumulator with negative and positive full scale $-f_{scale}$ and $f_{scale}-1$. Multiplication of $\Phi[n]$ by (π/f_{scale}) yields the true phase in radians, though this is unnecessary in pre-normalised LUT addressing. Underflow or overflow of $\Phi[n]$ implements modulo-2π accumulation. A requirement is that $F[n]$ is pre-multiplied by (π/f_{scale}).

7.4 Efficient Sine Calculation

The efficiency of the phase-accumulator oscillator rests on the choice of algorithm for computing $\sin(\Phi[n])$. Several options are reviewed in this section.

7.4.1 Taylor's Series

$$\sin(\theta) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \ldots (-1)^{n-1} \frac{\theta^{2n-1}}{(2n-1)!} \quad \text{where} \quad -\frac{\pi}{2} \le \theta \le \frac{\pi}{2}$$

$$= (((\frac{-\theta^2}{7!} + \frac{1}{5!})\theta^2 - \frac{1}{3!})\theta^2 + 1)\theta \quad \text{for } n = 4$$

(7.13, 7.14)

The commonest numerical method for calculating $\sin(\Phi[n])$ is the Taylor's series expansion. It converges quickly to any required accuracy. The formula in eqn. (7.13) can be factored using Horner's algorithm to the form of eqn. (7.14) to optimise computation; for n terms, $n+1$ multiplies, $n-1$ additions and $n-1$ constants are needed (Weltner et al, 1996). An extra multiply is required to denormalise $\Phi[n]$ or additional arithmetic to implement true modulo-2π accumulation of $\Phi[n]$. Only the first and fourth quadrants are covered: the other two can be generated by a reflection.

For a digital audio oscillator, increasing n reduces the distortion manifest as a spectrum of unwanted harmonics. Fig. 7.6 plots n versus the maximum error in dB's between the nth-order Taylor approximation and $\sin(\theta)$ which is to be found at $\theta=\pi/2$. A value of $n=4$ relates to an oscillator with a lower bound on S/N of 76dB; sufficient fidelity for music synthesis. The cost is 5 multiplies. It is an expensive method for computing $\sin(\Phi[n])$ but attractive from one perspective in that it is a continuous approximation: the characteristic 'phase jitter' of LUT algorithms at very low frequencies is absent as discussed in the next section.
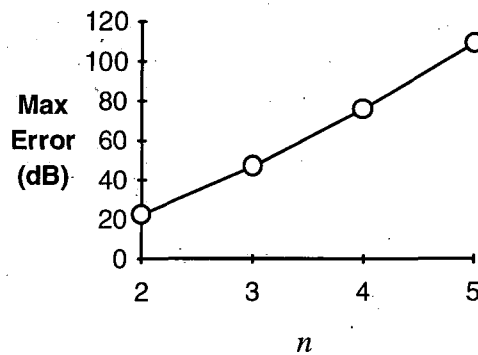
Figure 7.6   Max Error (dB) of nth order Taylor's Series approximation for sin(θ)

7.4.2 Look-Up Tables

The Look-Up Table (LUT) approximation for $\sin(\Phi[n])$ is an extensively documented technique for phase accumulator oscillators (Tierney et al, 1971; Rabiner and Gold, 1975). An LUT of length 2^X samples describes one period of a unity amplitude sinusoid: the LUT is addressed by the X most significant bits of $\Phi[n]$, constituting $int(\Phi[n])$ with the residual bitfield representing the fractional distance $frac(\Phi[n])$ between adjacent LUT samples. However, function symmetry may be exploited to reduce storage. Only one positive quadrant of 2^{X-2} samples is required to generate the other three by reflections involving complementing the LUT address $int(\Phi[n])$ on the 2nd and 4th quadrants and complementing the LUT output in the 3rd and 4th quadrants. Therefore the sign bit is not stored but is derived from $int(\Phi[n])$.

Oscillator S/N is governed by two variables; (i) the number of samples 2^X in a complete cycle and, (ii) the stored sample wordlength N, excluding sign. The optimum configuration (Moore, 1977) is $N=X$ where worst-case S/N=6(X-2)dB. For a given value of X, noise cannot be reduced by satisfying $N>X$ because of phase quantisation introduced by the effective discarding of $\mathrm{frac}(\Phi[n])$: commonly known as 'phase jitter' which has a quasi-harmonic spectrum. There are ways to reduce this. Rounding $\mathrm{int}(\Phi[n])$ on the basis of $\mathrm{frac}(\Phi[n])$ (i.e. to the nearest sample) improves S/N ratio by about 6dB and requires an extra bit of resolution resulting in $N=X+1$ and S/N=6(X-1). Also "dither" can be used to whiten the phase jitter spectrum (see section 7.4.4).

7.4.3 Linear Interpolated Look-Up Tables

An alternative to a fine resolution "direct" LUT is to use a coarser LUT with linear interpolation between consecutive samples: high accuracy is possible as a single quadrant of $\sin(\Phi[n])$ is a continuous function within finite bounds (Moore, 1977). The computation has three stages:-

1. $a=\mathrm{LUT}[\mathrm{int}(\Phi[n])]$

2. $b=\mathrm{LUT}[\mathrm{int}(\Phi[n])+1]-a$.

3. $\sin(\Phi[n])=a+b\times\mathrm{frac}(\Phi[n])$

The optimum configuration is $N=2(X-1)$ where the worst-case S/N=12(X-1)dB. This method approximately halves X but requires two LUT accesses, an addition, a subtraction and a multiply step. It is attractive because, as X increases, savings in storage requirements become considerable. For instance, for S/N>80dB, and storing a single quadrant, a rounding oscillator has parameters $X=15$, $N=16$ constituting $(16\times2^{15})/4 =$ 16384 bytes whereas an equivalent interpolating oscillator has $X=8$, $N=14$ constituting $(14\times2^{8})/4 = 112$ bytes. As the first quadrant of a sine is convex, the error caused by interpolation always has a positive value. Adjusting each pair of points such that the mean error over the corresponding interval is zero simply removes a DC offset and does not alter the worst-case peak-to-peak error which determines worst-case S/N (Houghton et al, 1995).

There are two choices for VLSI implementation of an interpolated LUT oscillator; (i) a single LUT can be accessed twice and b postcomputed, or (ii) two LUT's in parallel where the second LUT contains the precomputed b. Given the small LUT sizes, the second option is more efficient because parallelism is exploited: such a solution is cited by Snell (1977) and Chamberlin (1980). The multiply step $\text{frac}(\Phi[n]) \times b$ represents the major overhead and, traditionally, an extra cycle of the $Ai[n]$ multiplier implements this but halves the throughput bandwidth of oscillator updates. However the maximum wordlength required to represent b, upon analysis, is $(N/2)+2$, which is also the optimum input wordlength for the interpolating multiplier and $\text{frac}(\Phi[n])$): determined from $\max(b)=2^N\sin(2\pi/2^X)$ when $\text{int}(\Phi[n])=0$. For the specification S/N>80dB, a $(14/2)+2=9$-bit 'b' LUT and multiplier is required for interpolation.
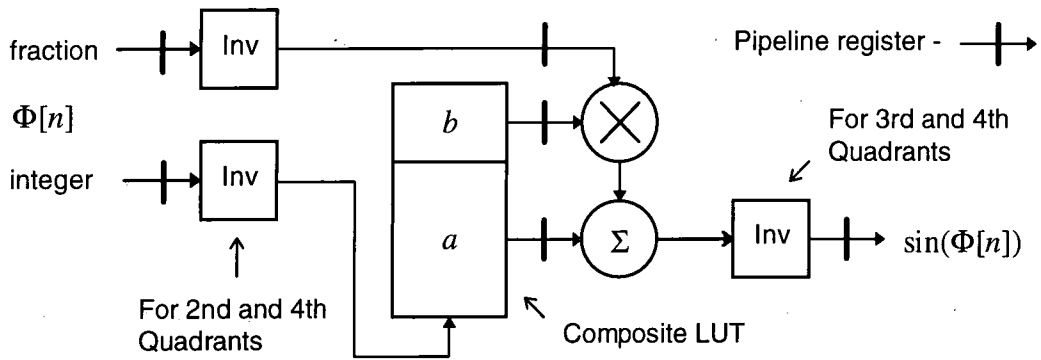
Figure 7.7   *Pipeline Form of Interpolating LUT Oscillator*

For VLSI implementation, a large rounding LUT presents two problems; (i) the silicon area required for S/N>80dB (16 Kbytes) is large and (ii) the propogation delay of ROMs increases with size posing the risk that it will determine an upper bound on TOB throughput which cannot be increased by optimisations elsewhere in the TOB. Therefore an interpolating oscillator appears to be a superior alternative. The option of two LUT's, one each for a and b, is fastest and may be pipelined with the subsequent multiplication / accumulation step as illustrated in Fig. 7.7. As both LUT's use the same address word - $\text{int}(\Phi[n])$ - extra efficiency is possible by coalescing both into a composite LUT. For S/N>80dB, the composite LUT size is thus $64 \times (14+9) = 64 \times 23$ bits. The chief overhead is the extra VLSI area required for the interpolation multiplier but since this quantity is $O(N^2)$ (Weste and Eshraghian, 1988) the area in this case is only $9^2/16^2=32\%$ of that

series of primitive angles constituted by Δ_i where $i=0..M-1$. For each primitive angle, the vector rotation can be expressed as matrix (7.15) with coefficients that are unity or such that multiplication can be implemented with binary shifts (attractive for VLSI implementation). Only the first and fourth quadrants are covered, the remaining two are generated by a reflection.

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & -\alpha_i 2^{-i} \\ -\alpha_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

$$\varepsilon_{i+1} = \varepsilon_i - \alpha_i \Delta_i$$

$$\Delta_i = \tan^{-1}(2^{-i}) \qquad\qquad (7.15..7.18)$$

$$\alpha_i = \begin{cases} +1 \text{ if } \varepsilon_i \geq 0 \\ -1 \text{ if } \varepsilon_i < 0 \end{cases}$$

where $x_0 = x, y_0 = y, \varepsilon_0 = \theta, -\dfrac{\pi}{2} \leq \theta \leq \dfrac{\pi}{2}, i = 0..M-1$

A side-effect of the CORDIC iteration sequence is that the vector is rotated with a constant gain K_M as a result of determinants in matrix (7.15) which are higher than unity. K_M is calculated by eqn. (7.19), which approaches an asymptotic value of approximately 1.647 for $M>4$. For applications where unity-gain rotation is required, normalisation by multiplying x_M and y_M by $1/K_M$ represents a hidden cost, although an alternative is to use another layer of normalising shifts and adds at each iteration.

$$K_M = \prod_{i=0}^{M-1} \sqrt{1+4^{-i}}$$

$$(7.19)$$

7.5.2 Application to Complex Oscillators

CORDIC is of interest because the rotation of a vector $A[n]+0j$ by $\Phi[n]$ is equivalent to $A[n](\cos(\Phi[n])+\sin(\Phi[n])j)$, thus eliminating the model of two sine calculations and two multiples in section 7.3. However, a number of modifications are required to make the algorithm suitable for a complex oscillator. First, the most significant bit (MSB) of $\Phi[n]_{MSB}$ is stripped leaving a residual field $\Phi[n]_{MSB-1..MSB-N-1}$ of N bits of which the MSB is the angle sign for two-quadrant CORDIC operation. (a fractional bitfield is left for

maintaining phase accumulation resolution). Full four-quadrant operation is then realised by initialisation of $\varepsilon_0 = \Phi[n]_{MSB-1..MSB-N-1}$ with the arctangent radix Δ_i pre-normalised by eqn. (7.20) as outlined in section 7.3, and x_0 conditioned by eqn. (7.21): implemented in logic by an XOR gate and a selectable two's complementer. As $y_0 = 0$, a simplification of stage $i = 0$ is possible which, by reference to eqn. (7.15), reduces to $(x_1, y_1) = (x_0, -\alpha_0 x_0)$, requiring only a single complementer.

$$\Delta_i = 2^N \tan^{-1}(2^{-i}) / \pi \tag{7.20}$$

$$(x_0, y_0) = \begin{cases} (A[n], 0) & \text{if } (\Phi[n]_{MSB}, \Phi[n]_{MSB-1}) \in \{(0,0),(1,1)\} \\ (-A[n], 0) & \text{if } (\Phi[n]_{MSB}, \Phi[n]_{MSB-1}) \in \{(0,1),(1,0)\} \end{cases} \tag{7.21}$$

CORDIC is a fixed-point algorithm. Input and output wordlengths are the same at N bits (identical for ε, x and y): truncation errors are introduced by multiplication by right shifts within the same wordlength (Hu, 1992b). Therefore the output has the same bit resolution as $A[n]$ which at low amplitude has low resolution and will yield a poor S/N. In contrast, imposition of $A[n]$ by an output multiplier has a constant value of oscillator S/N independent of $A[n]$, which is desirable for quantisation noise to be inaudible across a wide dynamic range of signal amplitudes: the multiplier is error-free as the output wordlength is the sum of the two input wordlengths and S/N is determined by oscillator quantisation effects alone.

A solution is to use floating-point rather than fixed-point representation. Mantissas are two's complement and range from $-f_{scale}$ to $f_{scale}-1$ where $f_{scale} = 2^{N-1}$. By normalising $A[n]$ to operate between $0.25 \times f_{scale}$ and $0.5 \times f_{scale}$, the output vector $x_M + y_M j$ varies in radius between $0.412 \times f_{scale}$ and $0.824 \times f_{scale}$, and may be denormalised by the normalisation factor of $A[n]$. Three barrel shifts are therefore required, with exponent extraction of $A[n]$. S/N will have a minimum value at $A[n] = 0.25 \times f_{scale}$ where the output resolution is poorest, giving a criterion for deriving design curves to assist in choosing a CORDIC design, parameterised by M and N. Normalisation for unity-gain rotation can be omitted as K_M from eqn (7.19) represents a modest constant gain offset for $A[n]$ of +4.3dB.

7.5.3 VLSI Implementation of CORDIC
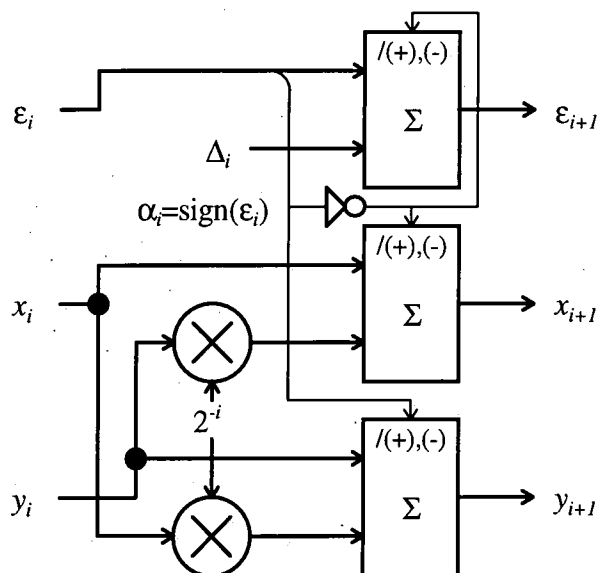
Figure 7.8   Dataflow for ith stage of CORDIC vector rotation

Fig. 7.8 illustrates the dataflow of the ith stage of a CORDIC iteration. Three additions / subtractions are performed in parallel at each stage. Two implementation strategies allow complexity to be traded off with performance. The first is serial computation. By adding three registers and feedback, calculation takes M clock cycles. The arctangent radix Δ_i is indexed from a table and multiplication of x_i and y_i by 2^{-i} is implemented by a combinational barrel shifter. Self-evidently, a serial CORDIC unit would represent an unacceptable bottleneck in a TOB that delivers one sinusoid update per clock cycle. In this case, LUT techniques are superior.

Data-dependence between stages precludes any performance gain by using a combinational approach, as in the array multiplier. Right-shifts on x_i, y_i make the lesser significant part of the sum for y_{i+1}, x_{i+1} dependent on the ripple-carry propagation to the most significant part of x_i, y_i. As $i \rightarrow M$ then carry-propagation must be complete for each stage before a valid result propagates to the next stage. However, the iterative characteristic of CORDIC makes it an ideal application for pipelining. An immediate by-product is that the barrel-shifters of the serial form can be omitted: the shifters are

Page 121

hardwired as an interconnect pattern unique to each stage, as is each Δ_i. Also, dynamic register circuitry required for pipelining adds minimal extra area (Ahmed, 1985).

In a pipelined CORDIC, individual stages are isolated with registers driven from a master clock. Calculation still has a latency of M cycles but the major gain is that throughput is limited only by the propagation delay of a single stage, equivalent to the ripple-carry addition / subtraction of two N-bit integers. Note that α_i (the sign bit of ε_i) is computed one cycle in advance. Pipeline latency represents a problem when computing a sequence of calculations where each successor is dependent on the result of its predecessor. However, the only post-processing operation of AS as expressed in eqn. (1.1) is accumulation and thus a CORDIC pipeline in a MAS context may operate at maximum throughput.

$$\{\text{Area of Serial } N \text{ - bit Multiplier } = 1.0\}$$

$$\text{Area of } N \text{ - bit Array Multiplier } = \frac{9(N-1)}{14} \qquad (7.22, 7.23)$$

$$\text{Area of } N \text{ - bit CORDIC Pipeline } = \frac{10N}{7}$$

A comparative study by Ahmed (1985) permits a quantitative evaluation of CORDIC. Eqns. (7.22) and (7.23) quantify the relative silicon area occupied by a Baugh-Wooley two's complement array multiplier and a CORDIC pipeline for an input wordlength of N relative to a bit-serial multiplier. NMOS technology is assumed with time for full addition (T_f) and for a logical AND (T_a) related by $T_f = 2T_a$. In terms of throughput, the multiplier has an upper limit on bandwidth of $1/(2(N+6)T_a)$ with carry lookahead in the last layer. For a CORDIC pipeline, this figure is $1/(2NT_a)$. However, as discussed in section 7.5.5, CORDIC actually requires a slightly longer internal wordlength to take into account truncation errors which offsets the faster speed. However, a CORDIC pipeline and an array multiplier have approximately equal throughput for large N.

7.5.4 Complex Oscillator Core with a CORDIC Pipeline

Fig. 7.9 illustrates the core of a phase-accumulating complex oscillator using a CORDIC pipeline resulting from an integration of the ideas of sections 7.5.2 and 7.5.3. $A[n]$

normalisation is achieved by a priority-encoder for extraction of the exponent controlling a barrel shifter ($2n$ input, n output bits) for extraction of the mantissa. The two output barrel shifters (n input, $2n$ output bits) use a delayed exponent to account for CORDIC pipeline latency of M cycles. Note that a single multiplexed shifter may suffice in implementation.
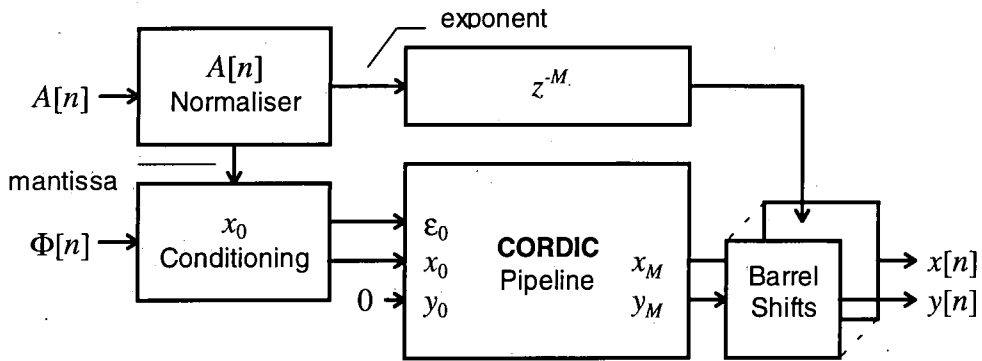
Figure 7.9   Schematic Diagram of CORDIC-based Complex Oscillator Core

7.5.5 Determination of CORDIC S/N Performance from M and N

In order to evaluate the performance of a CORDIC oscillator, a software simulation was written in 'C' under UNIX. The purpose was to derive an expression for worst case S/N in terms of wordlength N, and the number of stages M using a 'Monte Carlo' methodology similar to that used by Moore (1977) in his evaluation of LUT oscillators. Hu (1992b) provides a theoretical analysis of the quantisation effects of the CORDIC algorithm. However, the modifications of section 7.5.2 require validation and also, given the unusual use of CORDIC to for sinusoid computation for AS/MAS, it was considered appropriate to generate results within the actual application context.

Assuming f_s=44.1kHz (CD) and given values for M, N a CORDIC oscillator was run for 10^4 samples to generate the 88 fundamental frequencies of notes from A0=27.5Hz to A7=3520Hz. For each run, noise was calculated by subtracting the output of an 'ideal' oscillator operating at double precision, with both oscillators operating from the same $\Phi[n]$. The input vector (x_0,y_0) was set to ($0.25\times2^{N-1}$,0) as explained in section 7.5.2 to establish worst-case S/N. Both sine and cosine outputs were included in the computation. Finally, the worst case value of S/N for all 88 runs was recorded. The

results are plotted in Fig. 7.10 and it can be observed that for a given value of N, adding more stages (increasing M) causes a linear increase in S/N which converges to an asymptotic limit as $M \rightarrow N$.
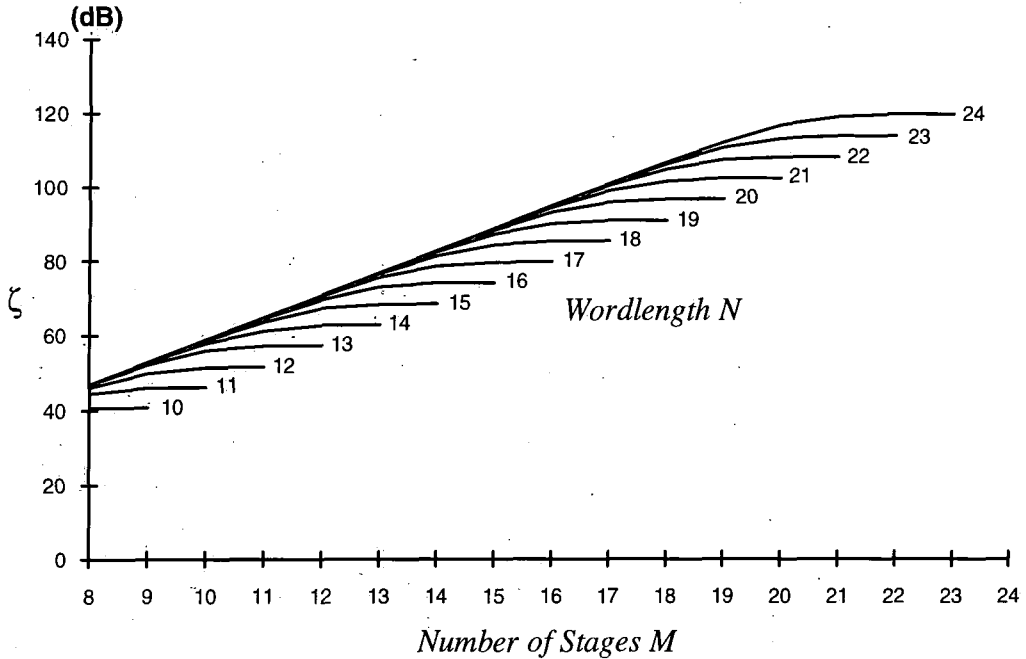
Figure 7.10   CORDIC Oscillator Performance

For a CORDIC oscillator of S/N=80dB therefore, $M=14$ pipeline stages with an internal wordlength of $N=18$ is optimal. This corresponds to >14 bits precision emphasising that each CORDIC stage adds an extra bit of resolution to the output sinusoid (S/N improves by 6.02dB/bit). The requirement that $N=18$ supports the formula proposed by Hu (1992b) that for M bit accuracy, $N \approx M + \log_2(M)$. Accumulated truncation errors are isolated in the 4-bit fraction which is discarded. Application of eqns. (7.23, 7.24) predict that the proposed CORDIC pipeline will occupy approximately $\times 2.7$ the area of a 14-bit Baugh-Wooley multiplier: Ahmed (1985) assumes M stages with $N=M$, in contradiction to Hu (1992b), and hence the approximation $N=16 \approx \sqrt{(14 \times 18)}$ is applied. However, additional area is required by the modifications of section is 7.5.4 which is dominated by the proposed barrel shifters.

7.6 Conclusions

From the survey, there are two efficient schemes for the realisation of a single-phase oscillator bank which satisfy the requirements of section 7.1; (i) the modified coupled form (recursive) and (ii) the pipelined linear-interpolated LUT (phase-accumulator). The former requires three multiplies (including imposition of $A[n]$) per sample, uses two state variables $(x[n], y[n])$ and phase is accumulated implicitly. The latter requires only a single variable for explicit phase accumulation ($\Phi[n]$) and a single multiply for $A[n]$, with a reduced wordlength multiply for interpolation and small LUT. Custom hardware is required to exploit its efficiency, but in such a form is more economic than the modified coupled form and hence is the recommended choice for VLSI implementation.

A complex oscillator bank for use with the PEF of section 3.7 is of similar computational complexity to its single phase equivalent. A modified CORDIC algorithm offers an alternative to two parallel instantiations of a pipelined linear-interpolated LUT. In spite of comparable performance, the relative VLSI design complexities will favour the simplicity of the latter approach as it is composed of a small set of standard components, which will be highly optimised. However, formulation of the CORDIC digital sine oscillator, hitherto uninvestigated for its application to AS, serves as an interesting contrast to the other techniques.

8. Specifying MAS Coprocessor Functionality

8.1 Overview

Having outlined the algorithm for MAS in terms of (i) the subband hierarchy of QMF-style filterbanks and (ii) a variety of candidate filterbanks with different performance / cost tradeoffs, the problem of mapping MAS to an architecture can be addressed. Design of an ASIC-based architecture for MAS represents the final section of research in this thesis. Fortunately, a headstart is provided by basing the design on the well-proven TOB of section 1.3.2 because MAS is an optimisation for accelerating the computation of AS in a multirate form of this architecture. In adopting this philosophy, the pitfalls of optimising within a narrow subset of AS applications are avoided (as discussed in section 2.3). For an ASIC to be economic, the application scope must be as wide and 'future-proof' as possible and fortunately, as outlined in section 1.1.1, these conditions are stong properties of AS and are preserved by MAS within certain constraints (e.g. latency, phase transparency). As discussed in Chapters 1 and 2, AS is less subject to 'paradigm shift' than alternative synthesis techniques.

AS is not computed in isolation but forms the basis of an SME (see section 1.3.2). Similarly, the output signals from AS are monoaural and require subsequent 'effects' processing. These processes will exist in classical AS implementations as software executing on industry-standard CPU's. Therefore, an ASIC-based MAS processor implements functionality that is inefficient to express in software, and is accurately termed a 'coprocessor' or MASC. An advantage of the pre-existing context into which MAS is transplanted is that aspects of the technique which are inefficient to map into the MASC (e.g. filterbank computation) can be mapped into software, leading to a hybrid implementation with a streamlined MASC that is more economic to realise. For a series of typical case studies involving the development of VLSI for music synthesis see Kahrs (1981), Kaplan (1981), Wawrzynek and Mead (1985), Mauchly and Charpentier (1987), Wawrzynek and von Eiken (1990) and Houghton et al (1995). A relevant parallel to the algorithm / architecture inter-relationships exploited during MASC specification is provided by Lee and Messerchmitt (1987a, 1987b).

This chapter starts with an informal systems analysis of the MASC context and the processes and dataflow of a MAS synthesiser are characterised. Then, identification of optimal MASC functionality is facilitated by mapping logical processes to physical processors, and partitioning the former between software and the MASC. SME functionality is not considered beyond the primitives required for MAS resource allocation. Next, a mechanism for Inter-Processor Communication (IPC) between the MASC and software processes is proposed in the form of a Shared Memory (SM). Efficient IPC is of equal importance to optimising the internal MASC architecture since MASC-SM IPC forms the critical section in the dataflow of a MAS synthesiser where control data bandwidth is at a maximum. An analysis of MASC-SM IPC leads to important insights into - and the specification of - the base-level data-structures and dataflow schemes by which both MASC functionality and throughput may be maximised. High-level and multirate constructs are superimposed on these footings in Chapter 9 leading to the objective of a high-level MASC design.

8.2 On the Economics of an ASIC-based MAS Implementation
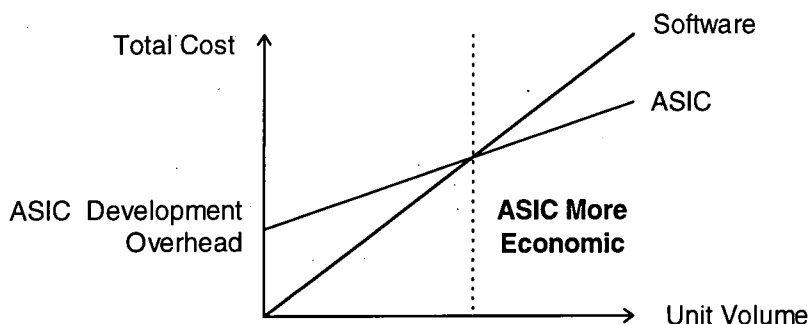
Figure 8.1   Comparison of the Economics of Software and ASIC Implementations

Figure 8.1 summarises the case for development of the MASC (in ASIC form). Consider the requirement for a high performance AS synthesiser. A software approach (c.f. FFT^{-1} in section 2.4) has a minimal hardware development overhead but the unit cost is high because expensive computer equipment is required (e.g. state-of-the-art workstations) to support the computational expense of AS. An ASIC approach (using e.g. a plug-in

MASC-based soundcard in a cheap PC) has high initial development costs, but a smaller unit cost to achieve the same performance requirement due to the refinement of TOB principles in the MASC (MAS is implementation-independent but uniquely compatible with a hardware-based solution as set out in section 1.3.2, in contrast to IFFT AS synthesis). Beyond a certain unit volume, an ASIC is the more economic choice. Also, as ASIC technology keeps pace with that used for flagship CPU's, it is reasonable to speculate that the cost gap can be maintained in the presence of technological progress. Generations of MASC may be envisaged with increasing speed, using the same architectural model. Also, the 'market-appeal' of AS, based on the properties of section 1.1.1, is likely to exceed the unit volume threshold required for an ASIC implementation to become economic.

8.3 Processes and Control Data in Real-Time MAS

8.3.1 Logical Process Pipeline

Before considering the mapping of processes to processors, it is advantageous to investigate the interaction of processes within a hypothetical MAS synthesiser. Fig. 8.2 illustrates how these processes form a logical pipeline through which input data - in the form of 'events' - is gradually transformed into a high-fidelity audio signal. The logical pipeline is synchronous in operation and, as discussed in section 1.2, must have a latency of less than T_{max} in order to satisfy the "hard" real-time response required of a live performance application. Pipeline synchronicity signifies that output data from one process may be consumed immediately as input by its successor: buffering is minimal and, furthermore, undesirable because it increases pipeline latency. Such pipeline structures abound in digital audio processing because they are the optimal way to perform sequential transformations on single-dimensional vectors of indefinite length such as digital audio signals (Higgins, 1990; Leiss, 1995). As control bandwidth is of critical interest to the AS debate, Fig. 8.2 also illustrates how this quantity may be perceived as changing along the length of the pipeline.

As discussed in section 1.3.2, the SME is represented by software executing on the main CPU of a MAS synthesiser which generates PWL envelope breakpoints from events in

real-time. Events originate from SME peripherals such as MIDI keyboards which generate discrete note on / off events, or more sophisticated transducer technology which transform the physical gestures of a musician into continuous time-varying parameters, which can be interpreted by the SME as AS metaparameters (see section 2.3). IPC from the SME to a logical 'Multirate Oscillator Bank' (MOB, c.f. the TOB of section 1.3.2) is therefore composed chiefly of a breakpoint stream, accompanied by auxiliary control signals related to states in the lifecycle of an oscillator (i.e. initialisation, update, termination). A single item of event information is transformed into a larger set of breakpoint data, hence there is an expansion of data bandwidth.
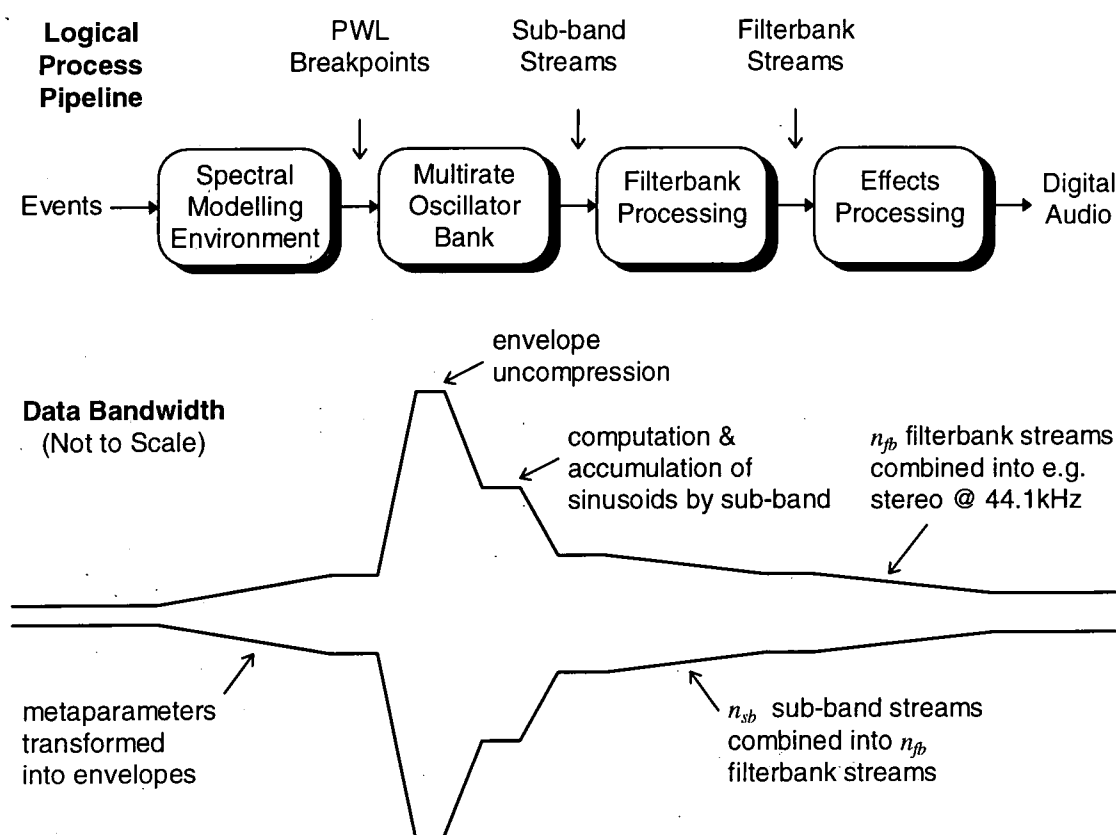
Figure 8.2   Processes and Control Data Bandwidth in a MAS Synthesiser

In the MOB, $A_i[n]$ and $F_i[n]$ envelopes for $1 \leq i \leq S$ oscillators are uncompressed in real-time, and phase $\Phi_i[n]$ is accumulated - as expressed in eqn (1.1) - creating the largest bandwidth in the pipeline in terms of arithmetic operations that must be computed in parallel. If N operations are required for an oscillator update, pipeline bandwidth is order(NS). In the TOB, control data proliferation is managed by exploiting data-

parallelism with replicated arithmetic units. Next, $A_i[n]\sin(\Phi_i[n])$ is computed and accumulated with previous sinusoids allocated to a specified subband and filterbank. In contrast, the bandwidth of this operation is order(S). After accumulation, bandwidth is order(n_{sb}) where n_{sb} is the number of subbands in the set of n_{fb} filterbanks where, generally, $n_{sb} \ll S$. During filterbank processing, the n_{sb} subband streams are further condensed into n_{fb} filterbank streams which are passed on to effects processing which generates the final digital audio stream in an industry-standard format e.g. CD.

8.3.2 Control Data Proliferation

The point illustrated by Fig. 8.2 is that data bandwidths throughout a hypothetical MAS synthesiser are at unexceptional levels except within the MOB where there is a proliferation in the quantity of control data caused by uncompression of $A_i[n]$ and $F_i[n]$ and computation of $A_i[n]\sin(\Phi_i[n])$. The oft-cited control problem of AS is attributable to this section of the pipeline alone and it is important to distinguish it from other sections in order to demonstrate that the problems experienced by classical AS implementations have an isolated cause which can be alleviated by optimisations such as MAS. To its advantage, MOB processing is a deterministic transformation and therefore a mapping to a static configuration of hardwired functional units in a MASC is an attractive option (c.f. the TOB). Two objective reasons for this approach are that (i) functional units are envisaged which are alien to standard CPU's (e.g. pipeline interpolated LUTs from section 7.4.3) and (ii) the fixed systolic topology of the TOB form within the MASC results in all functional units operating at maximum throughput with minimal computational redundancy unlike their sequencing in a CPU by an instruction stream.

The chief effect of MAS is to reduce control data proliferation in the MOB; the $E1$ benchmark defined in section 6.4.3 directly measures this performance improvement. A side-effect is the requirement for filterbank processing, and in relation, the higher bandwidth of output data from the MOB, as compared to classical AS, constituted by a number of digital audio streams totalling n_{sb} rather than n_{fb}, though subbands at depth $k > 0$ will be decimated thus minimising the bandwidth increase. MAS does not reduce the bandwidth of breakpoint data from the SME to the MOB, but this section of AS

processing is not perceived as requiring computational optimisation because PWL envelopes (see section 1.1.3) have proven the best practical representation for control data in terms of efficiency and transformability. However, a problem is posed for SME design in how such a representation is to be exploited, but such a consideration is predicated upon having sufficient AS resources which can be facilitated by MAS.

8.4 Mapping Processes to Processors

8.4.1 High-Level Multiprocessor Topologies
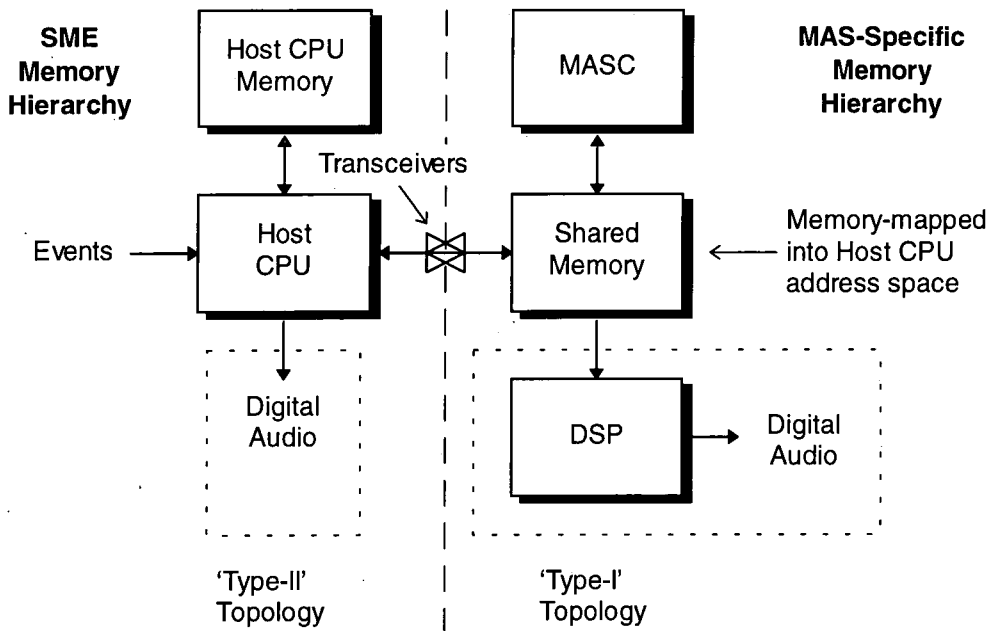
Figure 8.3   Architectural Forms of Multiprocessor MAS Synthesiser

Fig. 8.3. illustrates the two forms of multiprocessor architecture that are considered suitable for MAS, both of which incorporate a MASC. Mapping the logical pipeline of four processes in Fig. 8.2 to physical processors can proceed as follows. Most clearly, SME software is supported by the 'Host' CPU: the main microprocessor system in a MAS synthesiser. The MOB, as explained in the previous section, is mapped into the MASC because of its high compatibility with an ASIC implementation. However, filterbank processing poses a dichotomy because (i) it has a static dataflow topology during synthesis indicating that (like the MOB) an ASIC is a feasible option but (ii) prior

to synthesis it is desirable to configure filterbank topology (to expected partial frequency distributions), and select filterbank type (e.g. PM-FIR / PA-IIR QMF's from Chapter 4) according to permissible cost / phase transparency tradeoffs. Requiring such flexibility raises the control complexity of ASIC design. Note that this chapter is concerned with single-phase filterbanks as they have the greatest optimality. However, exploitation of complex representation (in conjunction with the PEF filterbank of section 5.3) for maximum functional transparency uses fundamentally the same high-level structures, modified at a low-level for the different data type.

As DSP's are optimised *inter alia* for digital filtration, mapping filterbanks into DSP software appears the most economical solution. Effects processing is also a DSP application and it is logical to assign both processes to a single DSP post-processor. Optimising compilers that can express the inherent parallelism of these processes will facilitate code that will execute efficiently on the specialised architectures of DSP's (e.g. using multiple data buses) making an ASIC expensive by comparison (Higgins, 1990). This form of MAS synthesiser is denoted the 'Type-I' form. An additional justification is that incorporating a DSP liberates Host throughput from burdensome low-level DSP tasks (for which its architecture is less highly optimised) so that the Host is devoted to higher level SME processing. However, the differentiation between classical CPU's and DSP's has diminished over the past decade due to technological progress (Freed et al, 1993) and Host throughput in a future MAS synthesiser is likely to be sufficient for all the processes in Fig. 8.2 excepting the MOB: hence the 'Type-II' form. To re-iterate, MOB throughput, as illustrated by Fig. 8.2, remains uniquely large and is best assigned to the MASC.

8.4.2 Interprocessor Communication via Shared Memory

The next feature illustrated by Fig. 8.3 is that IPC in the multiprocessor is envisaged as being via a Shared Memory (SM), which is single-port meaning that processors must obey mutually exclusive access. However, standard memories can be used in place of more expensive multiport examples which are, moreover, unnecessary as an analysis of IPC shall show. The justification for a SM is based upon the sophistication of Host-MOB IPC, which is clarified by considering the functionality required over the lifecycle of an

Page 132

arbitrary partial x in note-based AS. First, a free oscillator is identified, assigned to x and then added to the group of oscillators already allocated (if any) to a particular subband - dependent on $\{f_{min}(x), f_{max}(x)\}$ - in the specified filterbank. Afterwards, the SME requires access to the control parameters of x for synchronisation purposes, especially at note onset. Once executing, x requires feeding with breakpoints: a buffer is desirable so that a number of breakpoints can be deposited at note onset and the MOB is able to proceed with PWL envelope uncompression autonomously. At the end of its lifecycle, the oscillator for x requires de-allocation and labelling as a re-allocable resource. Therefore, direct (and short-term asynchronous) access to shared data structures is preferable to a synchronous message-based scheme which would impair both Host and MASC execution, create a bottleneck and also require complexity to handle the diversity of Host-MOB IPC (Burns and Wellings, 1989).

Two primary storage functions are fulfilled by the SM in (a) storing instantaneous oscillator state, that is the $A_i[n]$, $F_i[n]$ and $\Phi_i[n]$ accumulators and (b) buffering breakpoint queues for $A_i[n]$ and $F_i[n]$ for all $1 \leq i \leq S$ oscillators. A potential problem with SM configuration is that the space occupied by (b) can easily exceed (a) for even modest buffer sizes, increasing SM size above the lower bound required by (a). The most economic solution is single buffering which requires an instantaneous Host response to provide a successor PWL line segment once x completes each predecessor: the junction should be 'seamless' for high audio quality. As a consequence, SM storage is dominated by (a) and is small. Clearly, with a large value of S, unpredictable demands are placed on the Host due to "hard" real time constraints in excess of that necessitated by a music synthesiser (where a response $<T_{max}$ suffices). The alternative of storing a complete breakpoint set in the SM is wasteful because the SM must be of the highest speed in order to handle the control data proliferation of AS, and is therefore likely to be costlier per bit than Host main memory, which may be slower because it does not form such a critical bottleneck in the Host system. Hence only breakpoints that are required in the immediate future should be stored in the SM, i.e. of the order of T_{max}. This is also in accordance with the notion of real-time translation of SME events by metaparameters

into PWL breakpoint data, rather than an exclusive reliance on storing the complete spectral models of analysed tones.

It is useful to draw parallels between the SM in a MAS synthesiser, and a cache memory in a standard microprocessor system. The principle of *temporal locality* states that if an item *a* is referenced, then *a* is likely to be referenced again; soon. A related concept is *spatial locality* which adds that neighbours of *a* are also likely to be referenced soon. When an algorithm obeys these principles, a *memory hierarchy* is an efficient means for achieving a substantial speedup in execution on a serial architecture: for a background see (Hennesy and Patterson, 1990). The accumulator reference rate for each oscillator x is deterministic at $f_s(x)$, corresponding to temporal locality. Current envelope breakpoints for x are accessed on the same basis as are, ultimately, successor breakpoints in each envelope queue thus corresponding to spatial locality. The SM is therefore acting as a cache to the MASC, forming part of a memory hierarchy - external to that of the Host - that is specially designed to contain AS control data proliferation, of which the apex is caching within the MASC itself. However, a common foundation is on Host main memory and mass storage. Self-evidently, AS execution obeys both of the principles of locality and therefore a separate memory hierarchy is an effective way to isolate the high MOB bandwidths of Fig. 8.2 from interfering with SME execution on the Host. Fig. 8.3 also illustrates the inter-relationship of MASC and Host memory hierarchies.

The SM is memory mapped into a free portion of the Host's address space so that it is indistinguishable from conventional memory to Host software, facilitating random access to all MASC variables, which are stored exclusively in the SM. This form of IPC is suited to multiprocessor configurations where a large pool of data is shared by two or more processors (DeCegama, 1989) and such is desirable in a MAS synthesiser in order to maximise the transparency of MASC operation to the SME in the interests of application generality. Host-MASC IPC has the property of being short-term asynchronous in that breakpoint data may be written in advance of what is required by the MOB; the slack is taken up by breakpoint buffering. However, a long-term synchronisation mechanism using frames is required which is developed in the next chapter. Thus the Host and MASC are in a 'loosely coupled' master-slave configuration.

Clearly, the SM operates in parallel to the Host memory system and mutual isolation is necessary which is implemented by a bus interface with three-state buffers from the Host address bus and a transceiver for the corresponding data bus, with a bus arbiter mapped into the MASC. For low-overhead IPC, the Host must be able to access the SM with zero wait-state, thus blocking MASC SM access. However, referring to Fig. 8.2, the breakpoint IPC bandwidth envisaged from the Host is lower than the required MASC-SM bandwidth and therefore the margin of MASC throughput sacrificed by this means is an acceptable alternative to requiring a costly multiport SM: a justification is the excellent data compression properties of PWL representation as outlined in section 1.1.3. An analysis is provided in the next chapter.

8.5 Specifying MASC - SM Traffic

8.5.1 The Oscillator Descriptor

A common logical format for Host-MASC IPC is required in addition to the physical mechanism of SM as outlined in section 8.4.2. From an object-oriented perspective, oscillators are the only objects manipulated in AS. Therefore Host-MASC IPC is mediated via a single prototype SM data structure, termed an Oscillator Descriptor (OD), of which S instantiations express complete MOB state. The Host writes control data into each OD which is subsequently read and interpreted by the MOB in the MASC. An efficient OD design is essential for maximising MASC performance, and a chief determinant is SM width. With current microprocessor technology, data bus widths of 32-bits are considered conservative and 64-bits is a mature standard. SM width and that of the external MASC data-bus should reflect these conditions. However, the increase from a 32-bit to 64-bit MASC data-bus raises pinout and costs.

Fortunately the prototype OD has an inherent alignment on 32-bit word boundaries, with MASC OD accesses organised in word pairs. The latter property permits 'odd / even' interleaved access to a pair of 32-bit SM banks in parallel and provides twice the bandwidth of a single 32-bit memory bank. External latches are required to hold the address for one bank, which is in the data propagation state, when an address is presented to the other on the succeeding clock cycle. However, an SM composed of

DRAMs has such address latching by default. Therefore, the SM may be organised as non-interleaved 64-bit memory for the Host and, in contrast, as interleaved 32-bit odd / even banks for a 32-bit MASC which is functionally equivalent to the former, but with a halving in required MASC external data-bus width. At this level of discussion it is reasonable to leave SM organisation open as a future MASC design variable with 64-bit or 32-bit MASC non-interleaved data-bus formats as feasible options.
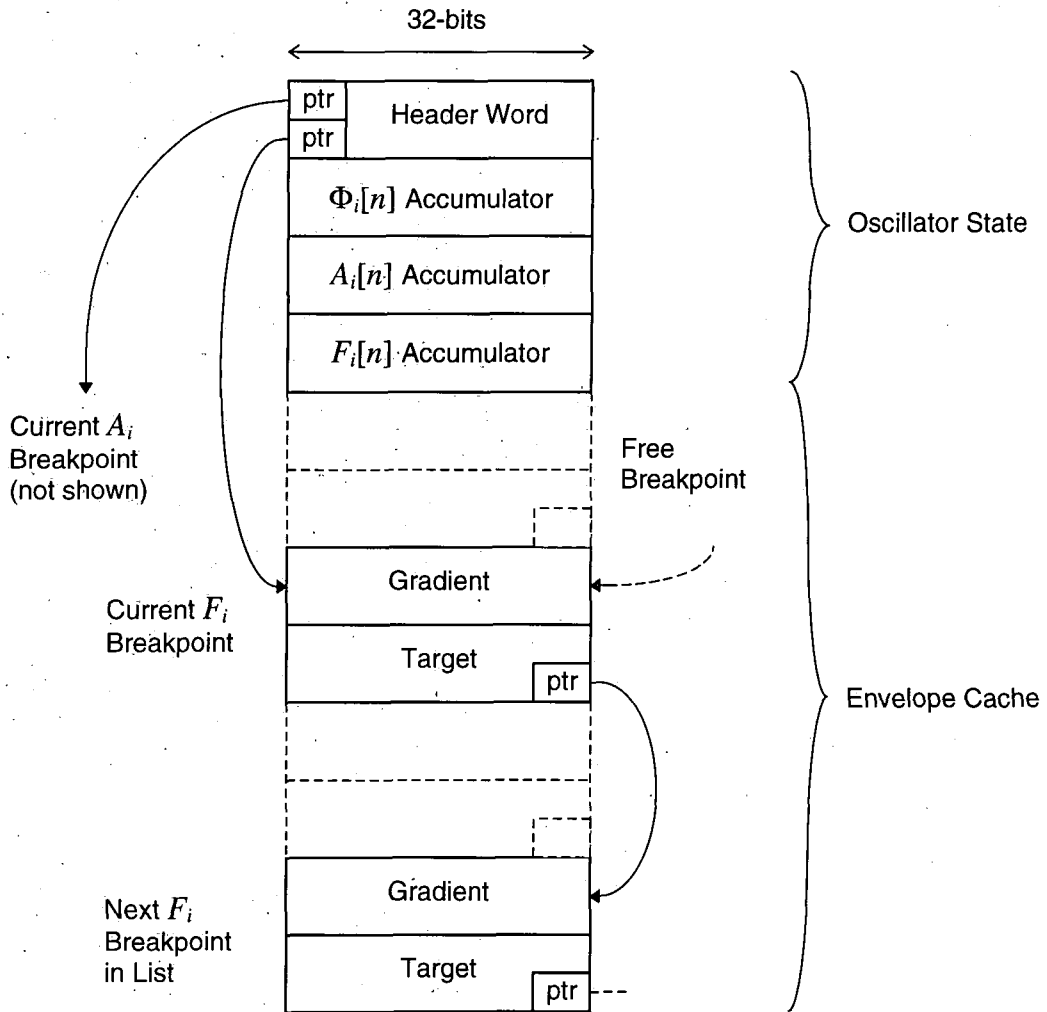
Figure 8.4   Oscillator Descriptor Organisation

Fig. 8.4 illustrates the internal data structure of the prototype OD. The first word is a header that contains miscellaneous status fields (tabulated in section 9.5.5) including two pointer fields referencing the current breakpoints for the $A_i[n]$ and $F_i[n]$ envelopes. The next three words are the phase accumulator $\Phi_i[n]$, and accumulators for $A_i[n]$ and $F_i[n]$. When an oscillator is updated by the MOB, the first operation is to load these four words

which express current oscillator state. Excess resolution is provided in the accumulators for (i) compatibility with standard integer wordlengths in Host software, (ii) high accuracy envelope uncompression and (iii) because the excess is not significantly greater than a lower desirable bound. To cite an example of a state-of-the-art TOB design (Jansen, 1991), 24 bits are used for $A_i[n]$ and 28 bits for $\Phi_i[n]$ and $F_i[n]$. One advantage of excess resolution is that it facilitates envelope line segments with near-zero gradients which can sustain an envelope at a perceptibly constant level for a specified short time interval; a feature which can be exploited for pre-skewing of envelopes in latency normalisation for the PA-IIR filterbank (see section 4.4.1), provided that the Host has computed the required skew *a priori*. An advantage of reduced sample rates in MAS is that an accumulator running at $2^{-k}f_s$ has an effective extra k bits of headroom over an accumulator of equal wordlength running at f_s, providing increased time resolution for low frequencies in agreement with Heisenberg's principle discussed in section 1.4.2.

Once oscillator state is initialised for an update, the current $A_i[n]$ and $F_i[n]$ breakpoints are loaded by de-referencing their pointers. Pointer wordlength is minimised by using the absolute SM address of the current OD as a base address so that a small workspace alone is in scope - termed the Envelope Cache (EC) - following the oscillator state variables, which has a maximum size determined by pointer wordlength. A speculative upper bound on EC size is 14 breakpoints, shared between $A_i[n]$ and $F_i[n]$, sufficient to store the attack transient and early sustain of most tones e.g. Grey's Trumpet of section 1.1.3. Thus 4-bit pointers are needed and overall OD size is a convenient 32 words. Breakpoints are naturally organised as word pairs where the first word is a two's complement integer representing the PWL line-segment gradient leading to the current breakpoint by which the relevant accumulator is incremented: no prenormalisation is used. The second word is the target value to which the accumulator is integrating. The incremented accumulator output is compared with the target and, if exceeded, is replaced by the target value. Upon breakpoint succession, the pointer in the header word is updated to reference the next breakpoint in anticipation of the next oscillator update. Clearly, target comparison is based upon the gradient sign; detection is required for overshoot in a positive gradient and undershoot in a negative gradient. Once $\Phi_i[n]$, $F_i[n]$ and $A_i[n]$ are incremented and the new value of $A_i[n]\sin(\Phi_i[n])$ computed, oscillator state

Page 137

is saved back into the OD in the order that it was loaded. The header word contains the updated pointers to current breakpoints.

A feature of the EC is the use of small-scale dynamic memory allocation. In breakpoints, the wordlength required to represent the target is less than that of the gradient because the latter requires a fractional field for the generation of line segments which have a gradient magnitude of less than unity. In contrast, target resolution need only match the smaller integer field that is extracted from envelope accumulators in the MOB which are, typically, 16 bits for $A_i[n]$ and 28 bits for $F_i[n]$ (Jansen, 1991). Therefore a small bitfield is free in the target word which can be used, to advantage, as the pointer to the next breakpoint: as stated earlier, 4-bits is considered sufficient. The principal reason for organising breakpoints into linked lists is that (i) the data is naturally organised in this way and (ii) there is minimal MASC and SM overhead in supporting lists and (iii) storage allocation in the EC is rendered much more flexible than if fixed tables were used. In consequence, a dynamic memory management overhead is placed on the Host. An elementary structure to employ with low Host overhead is a fixed-size ring FIFO buffer with the list arranged as a loop which would remain static over the lifetime of the oscillator. Division of the EC into unequal FIFO lengths for $A_i[n]$ and $F_i[n]$ reflects their different control bandwidths: $A_i[n]$ may be more active and take advantage of a longer FIFO. Alternatively, more sophisticated structures are facilitated such as a list terminating in a loop, for an attack envelope with periodic time-varying sustain which is executed autonomously by the MASC without Host intervention providing scope for enriching MASC functionality.

8.5.2 Interleaved Burst Processing

The proliferation of AS control data bandwidth discussed in section 8.3.2. is constituted in practice by the MASC-SM traffic required to update each OD in a round-robin schedule of S OD's with period $1/f_s$. Processing is entirely deterministic and there is scope for a full exploitation of SM bandwidth through a hardwired optimisation of MASC-SM traffic. A property which facilitates this is that AS processing, as discussed in section 8.4.2, obeys principles of temporal and spatial locality. In PWL envelope uncompression, the same breakpoint is used to generate a finite sequence of envelope

samples. Therefore oscillator state and current breakpoints may be *cached* by the MOB to generate a *burst* of output samples. Theoretically, no OD access is required until the next breakpoint is encountered. This is undesirable because the MASC is blocked to succeeding oscillators for an indefinite duration, conflicting with the real-time constraint that all S oscillators are updated at a fixed rate of f_s. However, caching for burst processing emphasises the inefficiency of multiple SM accesses to generate a single output sample. In a TOB, the bottleneck of a single data-bus is avoided by distributed state memories which permit an OD update each clock cycle. However, multiple data-buses mitigate against economic implementation of an ASIC-based TOB.

An elegant solution is facilitated by making MOB burst duration equal to that required by the SM-MASC traffic for a single OD update. Fig. 8.5 illustrates the burst processing scheme envisaged for the MASC which is based upon three principal operations:

1. Fetching of OD state from SM to initialise the MOB burst

2. The MOB burst

3. Saving of updated OD state back to SM

The current $A_i[n]$, $F_i[n]$ breakpoints addresses are contained in the OD header so that the latter must be fetched first. An assumption is made that line-segments associated with current breakpoints either outlast the burst or, alternatively, terminate precisely at the end, so that breakpoint succession occurs between scheduled bursts by a simple replacement of pointer in the header word with the next-breakpoint pointer from the predecessor breakpoint. Therefore, no additional fetches of successor breakpoints are necessary which can degrade an optimised schedule. As a consequence, however, breakpoints are quantised over a coarse time grid, at integer multiples of the MOB burst length, rather than at f_s. Such an imposition is not a disadvantage because breakpoints are naturally sparsely distributed over time, but a minimum resolution is desirable for accurate transient reproduction: for comparison, FFT^{-1} is quoted as quantising at multiples of 128 samples (Rodet and Depalle, 1992). A burden is placed upon the Host in computing breakpoint data such that envelope uncompression 'self-quantises' over the coarse time grid without violating MOB burst integrity. Such a burden is minimised by (i) excess accumulator resolution in the OD, in conjunction with the reduced sample

rates of MAS and (ii) by rounding-up the magnitude of desired gradients within gradient wordlength and relying upon overshoot or undershoot to effect self-quantisation. If an MOB burst is violated, a pragmatic strategy is to maintain the value of the violating parameter ($A_i[n]$ or $F_i[n]$) temporarily at the target of the new breakpoint until completion of the burst.

MOB Processing

OD_{i-1} Burst	OD_i Burst	OD_{i+1} Burst	OD_{i+2} Burst

MASC-SM Traffic

OD_i Read	OD_{i-1} Write	OD_{i+1} Read	OD_i Write	OD_{i+2} Read	OD_{i+1} Write

OD Accesses

Read:
Header$_i$	Oscillator State
$\Phi_i[n-1]$	
$A_i[n-1]$	
$F_i[n-1]$	
A_i Gradient	Current A_i Breakpoint
A_i Target	
F_i Gradient	Current F_i Breakpoint
F_i Target	

Write:
Header$_i$	Updated State
$\Phi_i[n]$	
$A_i[n]$	
$F_i[n]$	
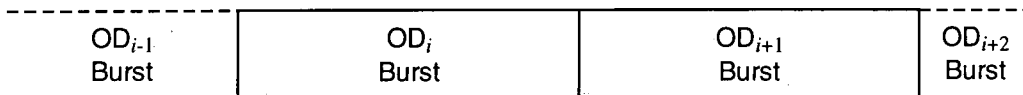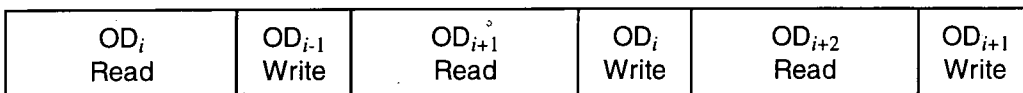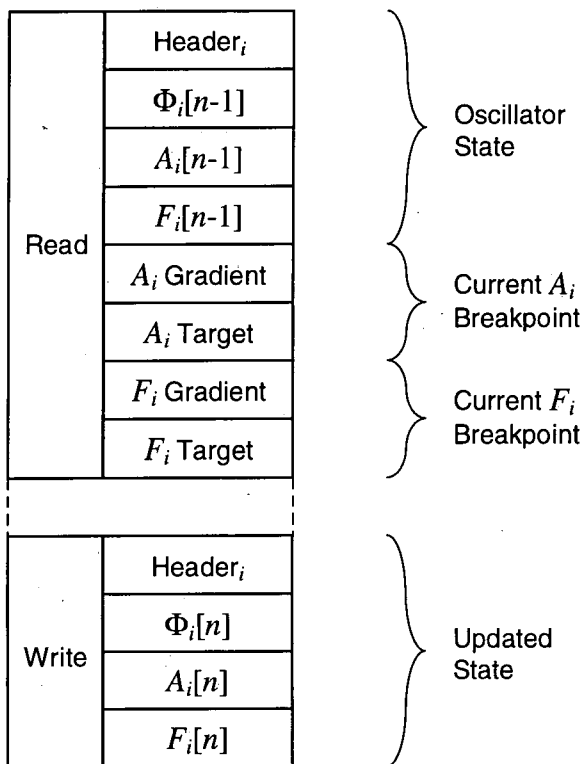
Figure 8.5   Interleaved Burst Processing

A requirement for optimal MASC efficiency is that MASC-SM traffic is transparent to the MOB which operates at maximum throughput with instantaneous context switches between oscillators. This is facilitated by *interleaving* (Lee and Messerchmitt, 1987a) the

three principal operations such that during each MOB burst, the updated OD from the previous burst is written back and the OD for the next burst is 'prefetched'. As a consequence, there are no SM idle cycles and MASC-SM bandwith is operating at its upper bound. In the MASC, Prefetch buffers are required to hold OD state for the next MOB burst which is transferred in a demultiplxed form into the MOB in a single cycle as in a TOB with parallel state memories. At the same instant, the final OD state of the last burst is transferred, in parallel into 'writeback' buffers for saving, in a multiplexed form, back to the SM. The architecture of the MOB detailed in the next chapter, is essentially the same as the TOB of section 1.3.2., except that state memories are represented by single registers holding the state of the current OD in order to generate a burst of consecutive output samples for a single oscillator. As illustrated in Fig. 8.6, (1) Host memory, (2) SM, and within the MASC, (3) OD prefetch and writeback buffers and (4) MOB registers constitute four ascending levels of the MAS-specific memory hierachy outlined in section 8.4.2.
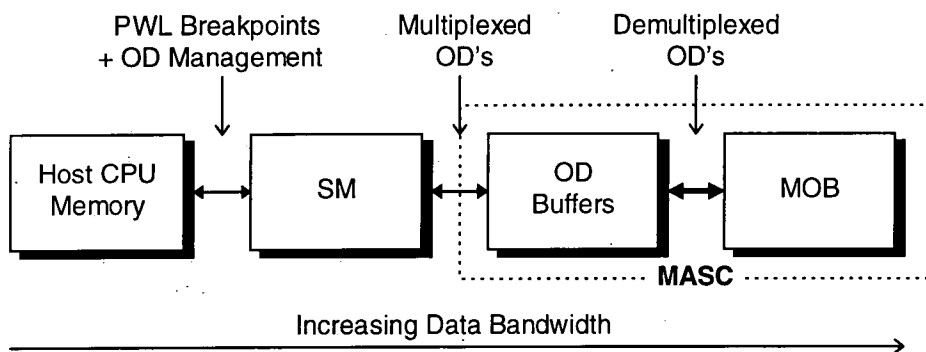
Figure 8.6   Model of MAS Memory Hierachy

9. Towards A MASC Design

9.1 Overview

In the last chapter, the notion of an Oscillator Descriptor (OD) is developed which offers an object-oriented insight into MASC execution. Also, the architectural context of a single external data-bus MASC accessing a single port Shared Memory (SM) under mutual exclusion with a Host CPU is developed, based upon the classical principle of memory hierarchy. Combination of these complementary principles leads to the idea of control data bandwidth optimisation in the MASC-SM data-bus via interleaved burst processing as summarised in section 8.5.2. These proposals are low-level solutions to the problem of handling high AS control bandwidth in a hardware implementation (as identified in section 1.1.2) and there remains the task, resolved in this chapter, of specifying the additional higher-level functionality for a complete MASC design.

First, the theme of section 8.5.2 is developed for a MAS context. The resulting 'frame-based' timing structure leads to scheduling, resource allocation and synchronisation solutions for realising the logical process pipeline of section 8.3.1. Much of the MASC functional specification is based upon observations of the relative control bandwidth of the constituent processes, illustrated qualitatively in Fig. 8.2. For a more quantitative estimate of MASC efficiency, the assignment of practical values to MASC design parameters leads to an understanding of how computational optimisation via MAS is manifest in the MASC, and of which design parameters have the largest impact on MASC performance. Finally, the dataflows and low-level logic diagrams for the major functional units within the MASC are documented.

9.2 MASC Functional Requirements for MAS

9.2.1 Supporting Multiple Sample Rates for MAS by Frame Scheduling

In the MOB burst processing scheme of section 8.5.2, the OD schedule period increases from $1/f_s$, as required by a TOB, to N_{burst}/f_s - where N_{burst} is the length of an MOB burst - in order that oscillators are updated at an aggregate rate of f_s. This longer schedule is

designated as a 'frame' and is the highest level timing construct in the MASC architecture, in contrast to the MOB burst, which is the lowest. Frames facilitate a simple and efficient transformation of the single sample rate MASC assumed in Chapter 8 into a multirate form for MAS. The frame period is set to T_{frame} as expressed in eqn (9.1) where the denominator is the sample rate of the lowest level of the subband hierarchy of depth K. Therefore the MASC is optimised for AS in the lowest subbands as illustrated in Fig. 8.5 where MASC-SM IPC is saturated. Higher aggregate sample rates for an oscillator x allocated to a subband at level $k:(0 \leq k < K)$ are generated by *extending* the MOB burst to $2^{K-k}N_{burst}$, that is to say multiples of 8,4 and 2 of N_{burst} for $k=0,1,2$ in a hierachy of depth $K=3$. Fig. 9.1 shows the impact of an extension of burst length to $2N_{burst}$ upon the single sample rate scheme illustrated in Fig. 8.5. which generates the correct aggregate sample rate for subbands at level $k=K-1$ i.e. the next subband series above the base level at K.

$$T_{frame} = \frac{N_{burst}}{f_s 2^{-K}} \tag{9.1}$$
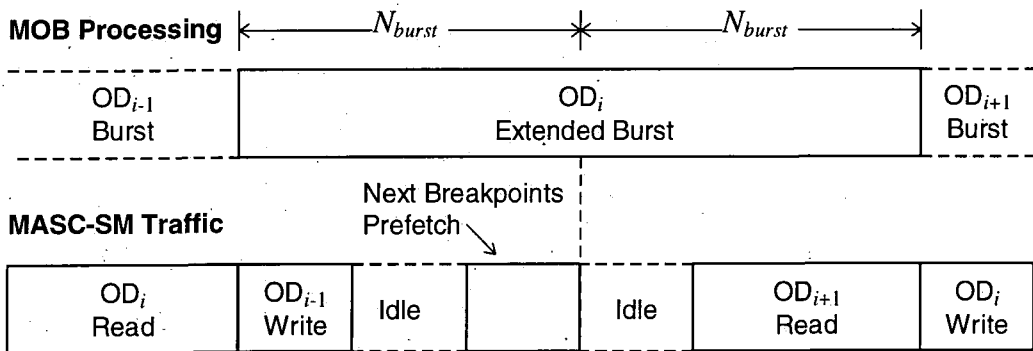
Figure 9.1   Extended Burst of $2N_{burst}$ for Subbands at Level $k=K-1$

Extended MOB bursts have the effect of introducing idle cycles into, otherwise, optimally interleaved SM-MASC traffic. However, to maintain a control rate proportional to subband sample rate in agreement with Heisenberg's principle (by breakpoint quantisation to integer multiples of N_{burst}) successor $A_i[n]$ and $F_i[n]$ breakpoints are prefetched every N_{burst} sub-division of the extended burst in anticipation of a possible succession. Only four words are accessed in place of the twelve required for a complete OD update and therefore the SM is idle for most of the extended burst

beyond the first N_{burst} sub-division which releases the SM for potential Host / DSP accesses without suspension of the MASC (a logical refinement would be to predict breakpoint succession so that breakpoint prefetch occurs only when necessary). A beneficial property of the frame is that it can support any distribution of MAS sample rates with the constraint that the net duration of all S bursts in the schedule must not exceed T_{frame}. Optimisation by MAS is reflected in the fact that extended bursts of higher subbands in the subband hierarchy ($k<K$) occupy a larger share of MOB throughput than non-extended bursts at the optimal level of K.

9.2.2 Burst Accumulation of Subband Streams

A concluding operation in classical AS is the accumulation of S sinusoids into a single output stream. In MAS, this operation is factorised into the accumulation of n_{sb} subband streams which are combined during filterbank processing into n_{fb} streams for effects processing: the MASC is responsible for the former operation. To ensure that accumulation introduces minimal roundoff noise, it is desirable that accumulation wordlength matches the output of the MOB $A_i[n]$ multiplier, which is the sum of the sine transform and $A_i[n]$ resolution, of the order of 16-bits apiece and therefore 32-bit accumulation is desirable. To integrate subband accumulation with interleaved burst processing as outlined in section 8.5.2 is straightforward if the OD schedule is coalesced such that all OD's associated with individual subband streams are executed consecutively in time; these groupings of OD's are termed 'minor runs'.

In place of the single register in the accumulator loop at the output of a TOB, the MOB has a FIFO of length N_{burst} so that arrival of the current MOB burst at the accumulator coincides with a recirculated burst comprising the sum of previous oscillators in the minor run. However, the accumulation burst must be initialised and, upon completion, communicated to the relevant subband and filterbank. To facilitate this, the OD header contains subband and filterbank addresses which permit detection by the MOB of a new minor run: signified by a change of subband address between succeeding OD's. When such an event occurs, the MOB blocks update of the new OD until the accumulation FIFO is flushed to buffers for reading by the Filterbank Processor (FBP) constituted by the DSP in the Type-I topology and Host in the Type-II: the best location for these

the first OD directly into the accumulation FIFO. To take into account extended bursts in MAS, the FIFO must support multiple accumulation burst lengths of $2^{K-k}N_{burst}$ for $k{:}(0{\leq}k{<}K)$. For $K{=}3$, this is facilitated by a cascade of four FIFO's in the accumulation loop of length 1,1,2 and $4{\times}N_{burst}$, with multiplexers prior to the last three which permit them to be switched into, and out of, the accumulation loop. Bypass of none, the last one, the last two, through to bypass of all three provides, in decreasing order, the requisite FIFO sizes. All FIFO's may thus be implemented as simple delay-lines. The accumulation structure is shown in section 9.5.4.

9.3 Scheduling, Resource Allocation and Synchronisation

9.3.1 Data Structures and Algorithms for Scheduling and Resource Allocation

The requirement for coalescing OD's into minor runs is non-trivial when the needs of resource allocation in MAS are considered. Each minor run is likely to be in a state of flux as new oscillators are added at the onset of a note, and removed upon its completion. To express the execution order of OD's, the schedule for each frame itself must be represented as a higher-level data structure referencing OD's at a lower level in the SM. A 'first-cut' solution is to execute OD's in the SM each frame as a sequential table ordered on ascending SM address. Clearly, a problem is that for optimal processing on the lines of section 8.5.2, OD's must be contiguous in the SM with minimal fragmentation: an unallocated OD will lead to wasteful stalling of the MASC for an entire OD cycle. However, adding a new OD to a compacted table requires block memory displacement to free the space required, consuming an extravagant number of SM cycles. Therefore a level of indirection is required such that the schedule order is not related to the absolute SM address of OD's.

The alternative of a sequential table of pointers to OD's has the same problems as a table of OD's, only that the overheads of defragmentation are reduced because pointers rather than OD's are displaced. Therefore organising OD's into a linked list (or rather loop, as the schedule is 'round-robin') is the most efficient data structure for representing the schedule (Moorer, 1982). The pointer from the current OD to its successor is contained in the header word and is a truncated form of its absolute SM address (see section 9.5.3).

Therefore, scheduling information is loaded transparently by the MASC in the same number of SM cycles as proposed in section 8.4.2. It is advantageous to align OD's on boundaries less than maximum OD size (determined by EC scope) so that smaller OD's can be used where necessary to economise on SM space. For efficient resource management in the SM, it must be configured such that there is a schedule of allocated OD's, and also a single 'free-space' list of unallocated OD's ready for initialisation by the Host and splicing into the appropriate point of the schedule, using standard linked-list algorithms (Tenenbaum and Augenstein, 1986). Upon termination, OD's are deallocated by removal from the schedule and are then added to the head of the free-space list pending future reallocation. So long as mutual exclusion for manipulation of the schedule loop is obeyed (outlined in the next section), these operations encompass all of the resource allocation functionality required between the MASC and Host. For comparison, Limberis and Bryan (1993) provide a case study of dynamic resource allocation in a conventional synthesiser.

9.3.2 Frame-Level Process Synchronisation
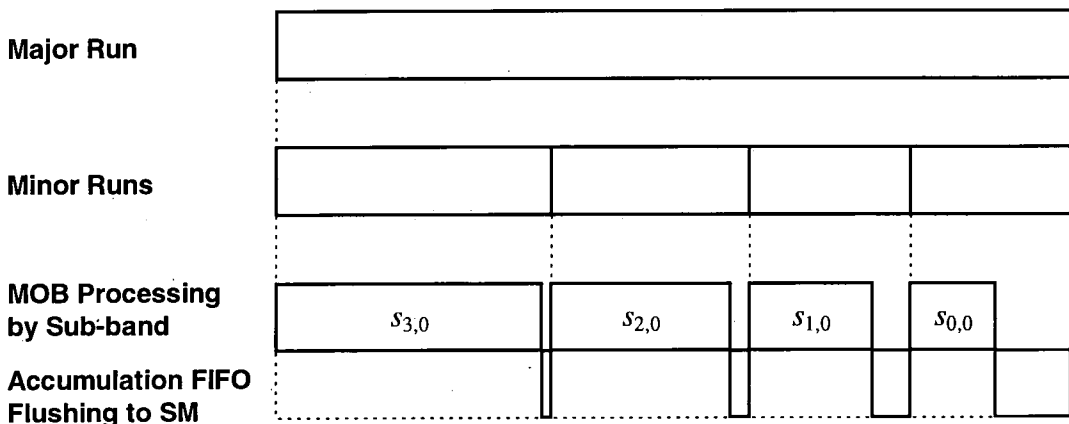
Figure 9.3 Minor / Major Runs in the OD Schedule

A linked-list schedule of OD's also facilitates correct synchronisation in (i) breakpoint IPC from the Host to the MOB and (ii) subband IPC from the MOB to the FBP in the presence of a dynamically modifiable schedule. The purpose of MASC synchronisation is to guarantee IPC between processes within an upper bound of T_{frame} which is chosen to

be sufficiently long to permit a margin of short-time asynchronicity between processes in the logical pipeline by exploiting the latency tolerance T_{max}. As stated previously, the schedule is partially ordered by coalescing OD's into 'minor runs' that pertain to each subband. A logical next step is to introduce a higher-level ordering by coalescing minor runs into 'major runs' that pertain to each filterbank. The organisation of a major run is illustrated in Fig. 9.3, with an (arbitrary) example of four minor runs for each subband level k of a filterbank of depth $K=3$: the effect of accumulation burst extension is shown (not to scale). The order of minor runs in a major run is not extrinsically important, but it is convenient to order the whole schedule on three keys of (*filterbank#*, and subband k and l). In consequence, a housekeeping overhead is imposed upon the Host in maintaining schedule order, which is minimised by the dynamic storage methods discussed in the previous section.

Filterbank Processing
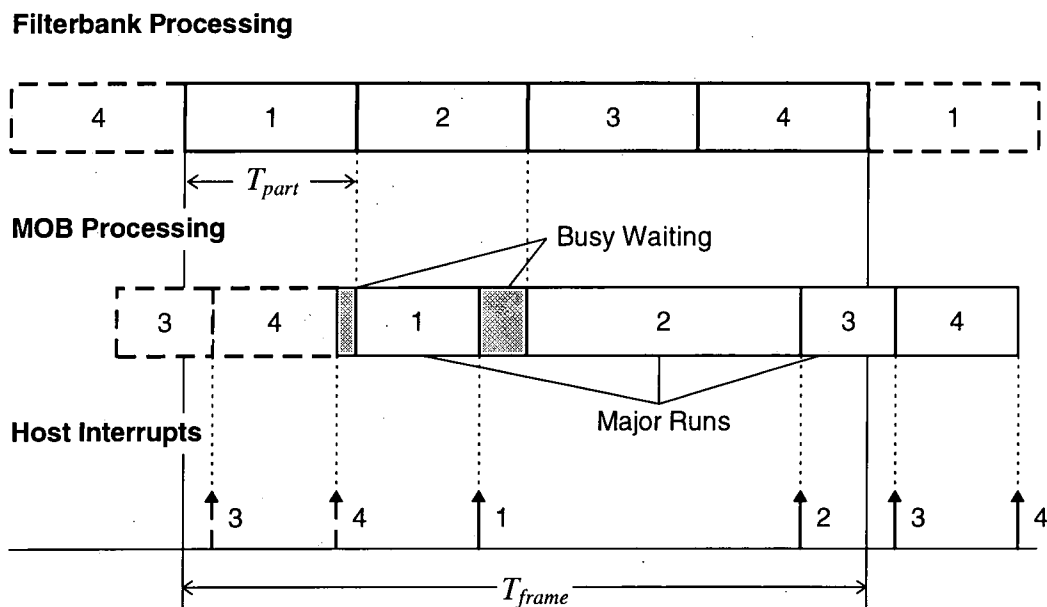
Figure 9.4   MASC Synchronisation for $n_{fb}=4$

The principal concept of the proposed frame-level synchronisation algorithm is that once ordered, MOB execution of the dynamic OD schedule is driven by the FBP which has, in contrast, a static schedule. Though the topology and number of filterbanks is reconfigurable, the configuration is assumed to be static during the 'run-time' of a MAS application. Fig. 9.4 illustrates an example for four filterbanks ($n_{fb}=4$) which can be extended to any value of n_{fb}. Each frame is divided equally up into n_{fb} partitions such that

filterbanks are executed in ascending order from the start of the frame: each jth partition denotes a time slot of duration T_{part} during which the FBP accesses only the SM buffers for filterbank j:$(1 \leq j \leq n_{fb})$. Filterbanks are executed depth-first by the FBP to generate a final burst of length $2^K N_{burst}$ which can be re-clocked via a FIFO at the required sample rate of f_s to the effects processor. Filterbank processing obeys principles of locality and may also take advantage of burst processing as far as FBP architecture permits. Filterbank topology is configurable implying a varying execution time but frame partitions are of equal duration. However, filterbank processing will occupy but a small margin of FBP throughput, and partitions are envisaged as being generated by a regular interrupt in the FBP system such that filterbank execution is completed within a modest interval at the start of each partition.

MOB execution of the major run for j must not overlap with the filterbank partition for j in order to maintain mutually exclusive access to SM buffers, otherwise there is a risk of data corruption. However, MOB operation can continue for other filterbanks. A simple algorithm to synchronise MOB processing to that of the FBP is (i) to order major runs in the OD schedule by ascending j and (ii) to allocate a synchronisation bit 'sync' to each major run, located in a reserved position of the OD header and which only has significance in the first OD of a major run which is signified when the MOB detects a change in filterbank address in the header. If $sync=1$, filterbank j has completed its SM buffer access for the current frame and set $sync$ thus so that major run j can be executed. However, if $sync=0$, filterbank j has not completed and the MOB enters a 'busy-waiting' state (Burns and Wellings, 1989) of repeatedly polling the header word (choice of an optimal polling frequency can release SM cycles for other potential operations). When the transition $sync=0 \rightarrow 1$ is detected, major run j is initiated and the header of the first OD is written back with $sync=0$ to reset the synchronisation mechanism. Fig. 9.4 illustrates that contiguous execution of major runs is permitted when no SM buffer conflicts occur. Two constraints are imposed for ensuring the integrity of the proposed frame-level synchronisation algorithm in that (i) the maximum length of a major run is $(T_{frame}-T_{part})$ and (ii) the net duration of a set of consecutive major runs which contains no repeated index j must be less than T_{frame}. Note that margins must be included for expected FBP and Host SM access bandwidths. Under typical conditions in note-based

music synthesis, these constraints are unlikely to be challenged, even in the presence of a dynamically configurable OD schedule that changes from frame to frame.

Host / MOB access to OD's under frame-level mutual exclusion is necessary during the initialisation and termination of oscillators. The MOB must execute only a completely initialised OD set for a new note so that oscillators commence execution correctly and partial envelopes are aligned in time. Also, the addition and removal of OD's requires temporary severing and manipulation of the schedule, violating its integrity and posing the danger that the MOB will reference invalid OD's, causing an MOB 'crash'. The mechanism for avoiding this state - also indicated in Fig. 9.4 - is for the MOB to send an interrupt signal to the Host upon completion of major run j indicating that there is an interval of time until the start of j in the next frame (a minimum of T_{part}) while j can be manipulated under mutual exclusion. At the end of the interval, schedule integrity must be restored by closing the reference loop. Frame-level conflicts are unlikely for OD update because the Host will interrogate the OD header for current status and supply breakpoints in advance of those currently queued. However, MOB interrupts provide the Host with a necessary mechanism for the real-time scheduling of OD updates. Contiguous MOB execution of a number of short major runs without 'busy-waiting' generates a burst of MOB interrupts which must be queued by the Host and processed sequentially. However, the response time to an MOB interrupt has loose tolerances as explained previously. Filterbank interrupts take precedence to MOB interrupts because the former is the driving 'consumer' process of the logical process pipeline.

An alternative solution is to eliminate filterbank partitions by arranging that the FBP copies all SM buffers with a burst read into its own local memory at the end of the frame whence filterbank execution can be scheduled arbitrarily during the next frame. The initial frame interval, devoted exclusively to MOB-SM traffic, is thus maximised. A problem with this approach is that subband streams pass via the FBP three times rather than once, as occurs in the scheme of Fig. 9.4. A hardware DMA module provides a potential solution. However, transparent DMA transfer using redundant FBP memory cycles would lengthen the SM read burst for which mutual exclusion with the MOB must apply, thus reducing the initial frame interval. An additional problem is that extra DMA hardware increases costs. Therefore the algorithm of Fig. 9.4 was developed which

permits random access by the FBP to SM buffers for the filterbank pertaining to each partition under mutual exclusion with the MOB. No extra hardware is required, FBP throughput is not degraded by unnecessary traffic and MOB-SM access is maximised as in the case of a burst read of SM buffers by the FBP, provided that the specified constraints on major run length are obeyed.

9.4 Quantitative Aspects of MASC Implementation

9.4.1 On the Interdependency of MAS Latency and MOB Burst Length

A prototype MASC can only be considered after the attribution of judicious values to the design parameters of K, N_{burst} and T_{frame}. It transpires that eqn. (9.1) is a fundamental relationship which governs how the inherent latency of MAS is traded off with a lower MASC unit cost. The first observation to make is that MOB clock rate may be higher than that for MASC-SM traffic because the former is encapsulated in VLSI and the latter is restricted by SM speed and the electrical properties of the external data-bus (e.g. capacitance). Therefore, N_{burst} is not directly dependent upon the number of SM cycles required for an OD update (six 32-bit word pairs as specified in section 8.5.1); a fact that has a major role in MASC efficiency because a high internal clock rate results in a costlier implementation because a faster VLSI technology is necessary.

From a consideration of the proposed MASC synchronisation algorithm of section 9.3.2, the maximum delay between a MOB interrupt to the Host upon completion of the major run for filterbank j and processing of the accumulation bursts from j by the FBP is $2T_{frame}$ as constituted by the eventuality of a short major run for j that is busy-waiting on filterbank partition j. Filterbank latency $T_{fb}(K)$ from eqn (4.7) must be added and also the delay in Host processing between the arrival of an event to the SME and its translation into MAS parameters, which is a maximum of about T_{frame} when OD schedule update by the Host is on a frame-level MOB interrupt basis (see section 9.3.2) giving a total (worst-case) system latency T_{tot} expressed in eqn (9.2) which emphasises the frame-level synchronicity of the logical process pipeline of Fig. 8.1. Effects processing is not considered because the signal processing applications involved are executed fullband and

there are none of the relatively high latencies associated with frame-scheduled multirate operation.

$$T_{tot} = 3T_{frame} + T_{fb}(K) \qquad (9.2)$$

On this basis, the optimum value of N_{burst} is determined by combining eqns. (9.2) and (9.3), which includes the latency tolerance T_{max} from section 1.2.2, to give eqn. (9.4). For example, if T_{max}=20ms, K=3 and f_s=44.1kHz and PM-FIR QMF filterbanks with Δ_f=0.1 as justified in section 6.4.4 are used, which have a stage latency of M=23 samples giving $T_{fb}(K)$=3.65ms @ f_s=44.1kHz via eqn (4.7), the result is N_{burst}=30. It is desirable for simple VLSI implementation that the clock frequency ratio between the MASC-SM data-bus and MOB is an integer power-of-two: as traffic on the former is organised as 6 word-pairs per OD, setting N_{burst}=4×6=24 is a sensible choice. T_{frame} must be recalculated by applying eqn. (9.1) with the practical value of N_{burst}. Despite a consequent reduction throughput from quantising N_{burst} thus, lower latency is a positive side-effect: with N_{burst}=24, T_{frame}=4.35ms resulting in T_{tot}=16.7ms via eqn.(9.2). Note that in this case, $T_{fb}(K) \ll T_{tot}$ showing that latency in MAS is attributable more to the exigencies of MASC optimisation than filterbank delay.

$$T_{max} \geq T_{tot} \Rightarrow T_{max} \geq 3T_{frame} + T_{fb}(K) \Rightarrow T_{frame} = \frac{T_{max} - T_{fb}(K)}{3} \qquad (9.3)$$

$$N_{burst} = \text{int}\left(\frac{f_s(T_{max} - T_{fb}(K))}{3(2^K)}\right) \qquad (9.4)$$

9.4.2 Quantifying Expected MASC throughput

An approximation of the maximum number of OD's that can be scheduled per frame by the MASC (if all are allocated in subbands at level K) is given by eqn (9.5). An interesting feature is that there is no K term indicating that max(S) is independent of multirate operation because the quantity is bounded by the MASC-SM bottleneck rather than MOB throughput. The effect of K is manifest in eqn. (9.4) in that N_{burst} is reduced

by a substantial amount (there is an approximate proportionality $N_{burst} \propto 2^{-K}$ if $T_{fb}(K) \ll T_{max}$) which is desirable for two reasons because (i) MOB clock rate is directly proportional to N_{burst} thus permitting the use of cheaper VLSI technologies for MASC implementation and (ii) the range of sample rates provided by MAS obeys Heisenberg's inequality (through extended bursts as discussed in section 9.2.1) which is not provided in a classical AS version of the MASC (i.e. $K=0$) which only has frame-level breakpoint quantisation c.f. FFT^{-1} (Rodet and Depalle, 1992). For $K=3$, the MOB operates at $1/8^{th}$ the rate required at $K=0$, with the limitation that $\max(S)$ is obtainable only in subbands at level K by exploiting the standard *a priori* MAS parameters of $\{f_{min}(x), f_{max}(x)\}$.

$$\max(S) \cong T_{frame} \frac{(\text{MASC - SM data - bus clock freq})}{(\text{no. of SM accesses per OD})} \qquad (9.5)$$

It is stated in section 6.4.4 that there is strong evidence for an optimal value of $K=3$. The insights into MASC implementation in this section also support this point of view. For comparison, setting $K=4$ and using the PM-FIR QMF stage prototype for $K=3$ of section 9.4.1 results in $T_{fb}(K)=7.82$ms making $N_{burst}=11$ by eqn. (9.4). A practical value of $N_{burst}=12$ is better for the reasons given earlier. Filterbank cost is increased causing extra processing by the FBP. Quite apart from a severe degradation of subband hierachy caused by the logical exclusion of deadbands, a higher $T_{fb}(K)$ reduces T_{frame} which, in turn, reduces $\max(S)$ via eqn (9.5). The benefit of the modification is that MOB clock rate is halved such that it is commensurate with the MASC-SM data-bus. However, this appears to be a false economy because higher clock rates are permissible in the MASC: this latter facility is not exploited and the system becomes bound by MASC-SM bandwidth. At $K=3$, MOB clock rate is twice the MASC-SM data-bus clock rate assuming a non-interleaved 32-bit SM. It appears reasonable to conclude, though it is desirable to reduce MOB clock rate, that beyond a certain limit economies from using MAS diminish. $K=3$ is evidently a good operating point.

Consider a hypothetical example of a complete MASC-SM system. Using memory devices with total access times of <100ns (e.g. high speed DRAM's) in an interleaved 32-bit SM (requiring 12 cycles for an OD update) with a data-bus cycle time of 50ns

(20Mhz) and T_{frame}=4.35ms (using the example in section 9.4.1) the result of applying eqn. (9.5) is max(S)≅7256. In this instance, MOB clock rate is 40MHz with N_{burst}=24. Such a figure constitutes AS resources in excess of those required by typical applications (e.g. those discussed in Chapter 6) and raises doubts about the utility of a non-interleaved 64-bit SM scheme as discussed in section 8.5.1. Optimal SM size is the product of max(S) and the size of the prototype OD (aligned at 32 words as discussed in section 8.5.1), plus space for MOB-FBP buffers (see section 9.4.3). In this example, an SM of 907kBytes, rounded to 1MByte, is indicated. However, max(S) is an upper bound and, in typical applications, allocation to subbands at level $k<K$ reduces max(S) which is compounded by FBP and Host SM bandwidth requirements (see section 9.4.3) indicating that there will be a large proportion of SM free-space.

9.4.3 Quantifying the SM Bandwidth Requirement of Host / FBP IPC

Host-SM traffic, which ultimately governs MASC operation, is difficult to quantify. However, some observations can be made. The PWL breakpoint set for an AS tone is usually highly compact (see section 1.1.3). At note onset, the major operation is the initialisation of OD's with sufficient breakpoints in the EC's for the immediate future (i.e. next few frames). So as not to degrade MASC performance more than strictly necessary, Host-SM accesses should be minimised. Also, Host-SM access is more complex than to main memory and may be slower, because of the necessity to suspend MASC-SM traffic beforehand. Host-SM traffic is characterised by bursts of control data upon OD initialisation superimposed upon a steady bandwidth of breakpoints to scheduled OD's. Because of the linked-list schedule, the cost of OD deallocation is minimal; a block of OD's is removed and appended to a free-space list by pointer manipulation of the starting and terminating OD's of the block, regardless of its size.

OD initialisation occurs on the precondition that the MASC can schedule the new OD's in which state there will be redundancy in MASC-SM bandwidth spent 'busy-waiting'. Host initialisation of an OD set can exploit this spare bandwidth without corrupting the processing of scheduled OD's by the MASC because of sub-frame-level asynchronicity between the MOB and FBP as indicated in Fig. 9.4. This is another desirable property of frame-level scheduling, apart from the ease of implementing multirate operations by

extended bursts (see section 9.2.1). In the eventuality of MOB-FBP frame-level synchronicity being corrupted by prolonged Host-SM access, the OD schedule will resynchronise within T_{frame} so long the constraints of section 9.3.2 on major run length are satisfied: FBP / MOB mutual-exclusion to SM buffers may be corrupted for a frame causing a 'glitch' in filterbank outputs. However, an efficiently designed Host-SM interface and MASC OS kernel for the SME can minimise this risk by a respective minimisation of MOB suspension and Host-SM access by using resource allocation algorithms dependent on Host-based images of the SM data structure organisation.

Quantifying the bandwidth of MOB-FBP traffic is also imprecise as filterbank number n_{fb} and topology are reconfigurable making the number of subband streams n_{sb} vary widely between different MAS applications. However, the simulations of Chapter 6 enable some speculative results to be derived. In section 6.4.2, $n_{fb}=8$ with identical instantiations of the prototype of Fig. 6.2: the input bandwidth to each filterbank b_{fb} is simply the sum of all constituent subband sample rates which is given by $b_{fb}=f_s+f_s2^{-1}+2 f_s2^{-2}+4 f_s2^{-3}=2.5 f_s$. Therefore at $f_s=44.1$kHz, $b_{fb}=110.25$kHz. As the MOB writes accumulation bursts to SM buffers which are subsequently read by the FBP, the total SM bandwidth occupied by MOB-FBP traffic b_{tot} is given by eqn. (9.6) which results in a requirement of $b_{tot}=1.764$MHz when samples (see section 9.2.2) and the MASC-SM data-bus are 32-bits wide. Considering that the MASC-SM data-bus is anticipated as operating in excess of 20MHz, the proportion of SM bandwidth sacrificed to MOB-FBP traffic is modest and below 10%. Therefore routing MOB-FBP traffic via the SM appears to be highly economic: a principle advantage is that no extra MASC pinout is necessary to support a separate FBP interface. It is relevant also to mention the total space SM occupied by MOB-FBP SM buffers as expressed by eqn. (9.7), which results in $n_{buff}=3840$ words (using the example of section 9.4.1 where $T_{frame}=4.35$ms): this is but 1.5% of the 1MByte SM proposed in section 9.4.2.

$$b_{tot} = 2n_{fb}b_{fb} \qquad n_{buff} = T_{frame}n_{fb}b_{fb} \qquad (9.6, 9.7)$$

9.5 Dataflow Aspects of MASC Design

In the light of the preceding analyses of MASC functionality, a systems-level MASC design can now be developed in terms of dataflows between functional units with the objective of identifying the primary internal architectural forms.

9.5.1 Systems-Level MASC Architecture
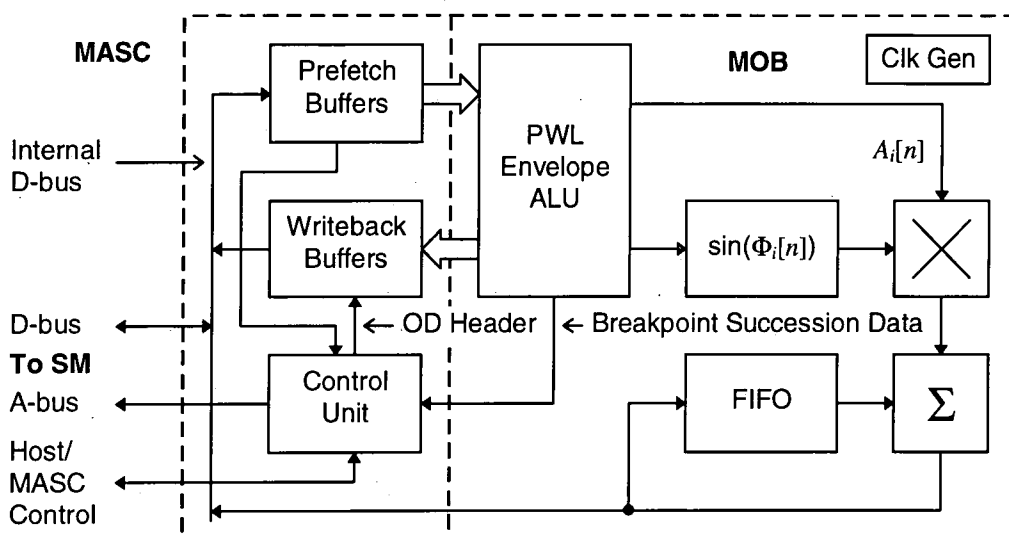
Figure 9.5   Schematic Diagram of MASC Architecture

Fig. 9.5 illustrates how the MOB dominates the architecture of the proposed MASC (Phillips et al, 1996b). When compared to the TOB of section 1.3.2, essential similarities between the dataflow topology can be identified: the distributed state memories of the TOB are lumped together in the MOB into a PWL Envelope ALU described in section 9.5.3. Three major differences are that (i) the MOB operates in a burst mode as specified in section 8.5.2, (ii) multirate operation is implemented by extended bursts in a frame timing structure as proposed in section 9.2.1 and (iii) computation of $\sin(\Phi_i[n])$ is envisaged as being via a linear interpolated LUT transform which is pipelined for high throughput as discussed in section 7.4.3. Non-MOB MASC functionality is centred around an internal data-bus whose activity is assumed to be identical to that of the external MASC-SM data-bus.

9.5.2 Prefetch and Writeback Buffers
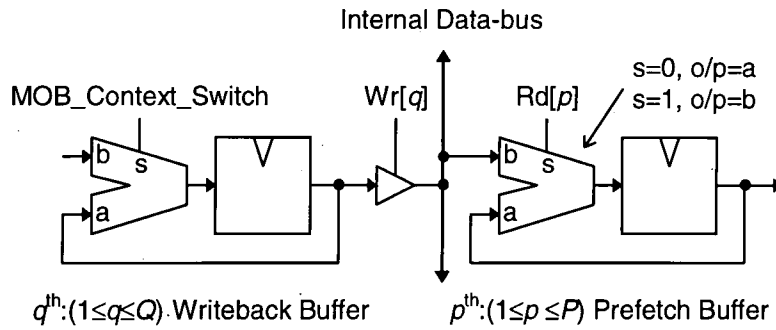
Figure 9.6   Prefetch and Writeback Buffers

Fig. 9.6 illustrates the logic for the Prefetch and Writeback Buffers. The header word also passes through the Control Unit via these buffers. During an OD read, $P=8$ words are read sequentially into the Prefetch Buffers, by sequencing Rd[1] through to Rd[P] as each OD datum appears on the internal data-bus. The OD is thus demultiplexed for parallel presentation to the PWL Envelope ALU and Control Unit. During breakpoint prefetch in extended MOB bursts, only those buffers pertaining to breakpoints are activated. Before an OD write, MOB_Context_Switch registers the final state of the current OD. Then $Q=4$ words (i.e. the header word and new accumulator values) are placed on the internal data-bus from the Writeback Buffers by sequencing Wr[1] through to Wr[Q] thus re-multiplexing OD data back to the SM. As discussed in section 8.5.2, these buffers constitute the third level of the MAS-specific memory hierarchy.

9.5.3 PWL Envelope ALU

The PWL Envelope ALU fulfills the two distinct function of (i) uncompression of $A_i[n]$ and $F_i[n]$ and (ii) integration of the phase accumulator ($\Phi_i[n]$) for sine transformation. For the former, two instances of the uncompression logic illustrated in Fig. 9.7 are required. When MOB_Context_Switch is high, a new OD is switched in upon the next MOB clock cycle (these input registers constitute the final fourth level of the MAS-specific memory hierarchy). Otherwise, when low, the accumulator is integrated by the gradient value and compared with the target value so that when undershoot or overshoot (depending on the gradient sign) is detected, the target value itself is substituted for the invalid accumulator value as specified in section 8.5.1. To signify breakpoint succession, New_Breakpoint_Ptr from the target and a Breakpoint_Overflow flag are supplied to the

Control Unit which handles the event. For breakpoint succession in extended bursts, a signal New_Breakpoint retains the current accumulator value, but registers the gradient and target of a new breakpoint from the Prefetch Buffers.
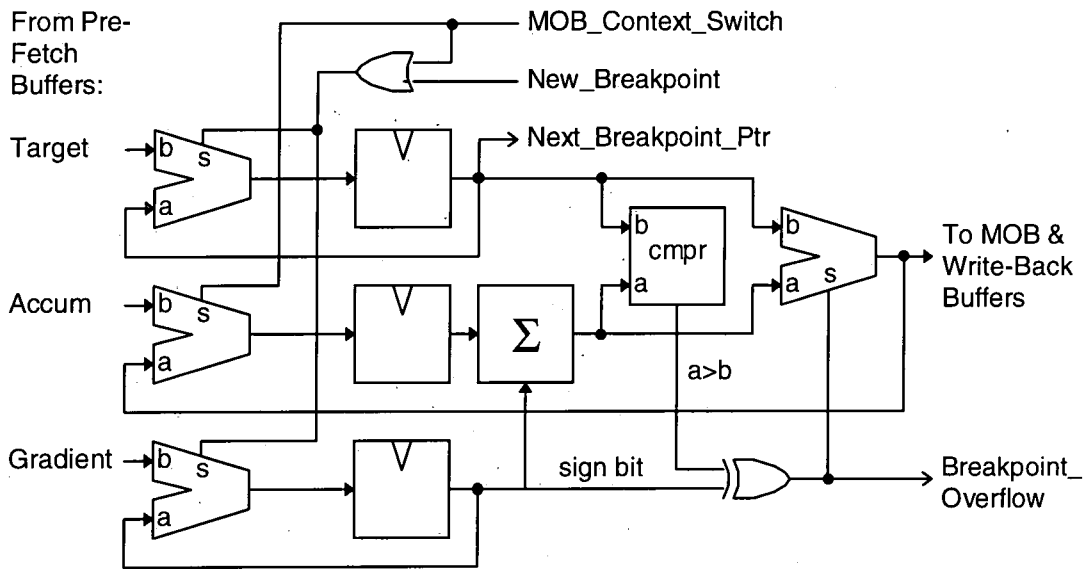
Figure 9.7   Dataflow for $A_i[n]$ and $F_i[n]$ Structures

Discrete integration of the phase accumulator ($\Phi_i[n]$) is rendered more complex by the necessity for trapezoidal integration according to the scheme of section 4.3.2 (bypassed for the case of a fullband OD allocated to $k=0$). Fig. 9.8 shows the resulting logic which has the same basic form as Fig. 9.7 minus breakpoint handling. $F_i[n-1]$ and $F_i[n]$ can be conveniently taken from, respectively, the accumulator register and multiplexer outputs in the $F_i[n]$ structure. 2^{-k} multiplication is effected by a simple three-line barrel shifter for $K=3$. Therefore, ignoring multiplexers, the critical path in the PWL Envelope ALU is (i) an addition and comparison (implemented by a subtraction) in the $F_i[n]$ structure and (ii) three additions in the $\Phi_i[n]$ structure: propagation must be complete in (i) for (ii) to begin. If carry look-ahead is used, and noting the properties of cascaded adders (c.f. array multipliers), throughput bandwidth should be commensurate with succeeding sections of the MOB pipeline. Internal pipelining in the PWL Envelope ALU is therefore not considered necessary as it carries the overhead of startup and flushing which precludes the requirement for instantaneous context switches between OD's.
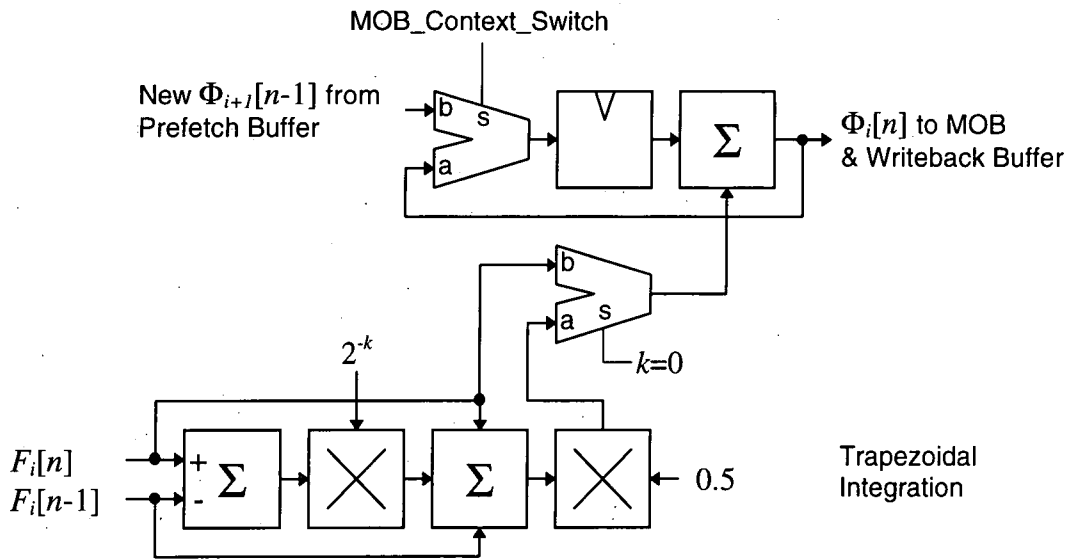
Figure 9.8   Dataflow for $\Phi_i[n]$ Structure

9.5.4 Accumulation Structure
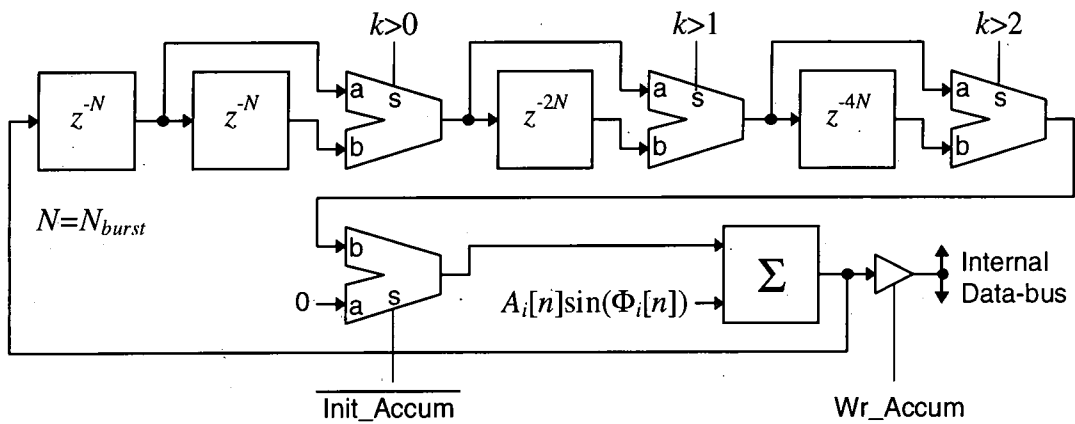
Figure 9.9   Accumulation Structure Dataflow

The accumulation loop FIFO is variable length following the scheme described in section 9.2.2 and is illustrated in Fig. 9.9. Each of the '$k>n$' signals represents a truth condition about the level in the subband hierarchy of the current minor run. An input multiplexer to the adder permits a zero addend when Init_Accum is low (implemented by AND gates) so that the FIFO may be initialised during the first MOB burst of a new minor run. When Wr_Accum is high, the accumulated output of the last MOB burst of a minor run is placed in the internal data-bus and written to the MOB-FBP buffers in the SM using an address burst provided by the Control Unit.

9.5.5 Header Word Processing in the Control Unit

Field	#bits	Comment
Next OD pointer	14	(16384 maximum)
Current $A_i[n]$ breakpoint pointer	4	(for the 32-word OD of section 8.5.1)
Current $F_i[n]$ breakpoint pointer	4	ditto
Filterbank Address	4	(16 maximum; c.f. section 6.4.2)
Subband Address k	2	($k=0..3$ for $K=3$)
Subband Address l	3	($l=0..7$ at $k=K=3$)
sync (from section 9.3.2)	1	for frame-level synchronisation

Table 9-1   OD Header Word Bitfield Assignment

The four tasks of the Control Unit are (i) interpretation and updating of header words, (ii) SM address generation, (iii) sequencing of internal MASC dataflows and (iv) SM access arbitration with the Host and FBP (combined in the Type-I topology). All but task (i) require an implementation-level of design but the major facets of functional behaviour are encompassed in this and the preceding chapter. The main operation in header word update is that of detecting a breakpoint succession from the PWL Envelope ALU and replacing the EC pointer for the relevant envelope with the next breakpoint pointer (derived from the target field of the current breakpoint) as outlined in section 8.5.1. The bitfields of the hypothetical 32-bit header word are tabulated in the Table 9-1. The 'Next OD Pointer' may be usefully aligned on multiples of 16 words as this permits a minimal OD with an EC of just 6 breakpoints, or longer by multiples of 8 breakpoints as desired. Filterbank and subband addresses (7 bits in total) may be used to address a MOB-FBP buffer pointer table of 128 words (see in section 9.2.2).

9.6 Review

The set of logic designs of MASC functional units provides an appropriate conclusion to the level of abstraction to which the MASC concept is developed in this thesis. A design for the Control Unit in Fig. 9.5 is the major omission. This is because it is dependent upon (i) the low-level organisation of the MASC-SM data-bus and (ii) final decisions about how scheduling and synchronisation may be implemented which can only be made within the context of MASC prototype development. However, the Control Unit will take the form of a state machine which can be synthesised from a lower-level behavioural (as opposed to functional) description of the MASC during simulation. For similar reasons, the mechanism for MASC suspension during Host-SM access is omitted; an efficient solution is to derive the dataflow clock from a higher frequency master clock (e.g. ×2) in a notional Clock Generator circuit for hazard-free gating of the dataflow clock.

10. Conclusions and Further Work

The main purpose for this thesis is an investigation into the systematic application of multirate DSP techniques to minimise the high computational bandwidth of classical AS in a TOB architecture. The rapid rise in CPU clock rates in recent years has meant that a software IFFT implementation of AS is perceived as rendering the custom hardware of a TOB obsolete. Simultaneously, interest in the TOB has declined because it is difficult to identify optimisations that do not compromise the generality of AS. However, it is argued that the chief functionality of AS is *the synthesis of musical notes* which have certain properties to which AS may be optimised, implying that TOB execution has high computational redundancy, thus leading naturally to the proposal of TOB optimisation by MAS. Unlike proposals for software IFFT synthesis, a noise model is excluded from MAS: justified on the basis that noise synthesis is not computationally intensive.

The idea of a time-invariant 'near-critical' sample rate is introduced for a time-varying oscillator x based upon *a priori* knowledge of its frequency trajectory during a finite lifecycle in note-based music synthesis. Exploitation of this property for sinusoidal oscillators in MAS creates two significant problems of (i) scheduling multiple sample rates and (ii) the interpolation of decimated sinusoids up to f_s to avoid quantisation noise. These are minimised by exploiting the *inherent hierarchy* of AS in that S sinusoids are summed into a final (multichannel) stream; the hierarchy is split and n_{fb} multirate filterbanks are introduced in the middle. A single filterbank interpolates many oscillators thus minimising interpolation overheads. Also, a small set of commensurate sample rates minimises scheduling overheads. The criteria for choosing a filterbank are (i) the 'goodness-of-fit' to expected oscillator distributions and (ii) how little the generality of AS is compromised. A binary-tree topology filterbank such as the QMF, when viewed as a *subband hierarchy*, is shown to provide a flexible and economic solution to both these desired aims.

Practical QMF implementation is shown to have two problems of (i) non-infinitesimal Δ_f and (ii) phase corruption through high-pass filtering. Economic FIR QMF designs have large Δ_f but are phase linear which is desirable for high functional transparency (to this

end, efficient frequency and latency normalisation schemes are proposed). A suitable polyphase-allpass IIR QMF design has minimal Δ_f but is phase non-linear. High Δ_f is undesirable as it reduces the efficiency of a subband hierarchy because oscillators must be promoted into higher subbands when Δ_f 'deadbands' are removed by 'logical exclusion'. Using the properties of complex signal representation, a novel filterbank - the PEF - is introduced where Δ_f is 'physically excluded' and which is demonstrated in simulations to possess total phase transparency. As it is envisaged that QMF filterbanks will be implemented in software, the choice of FIR *versus* IIR can be left to the end-user, depending upon the required phase transparency. To its disadvantage, the PEF introduces significant extra complexity into the MAS algorithm.

A hypothetical scenario of the synthesis of a complete symphony orchestra provides two benchmarks - $E1$ and $E2$ - which quantify, respectively, how control bandwidth and computation (in multiplication frequency) are both consistently reduced across the range of filterbanks options. An important fact to emerge is that optimal QMF filterbank depth appears to be $K=3$, especially when FIR implementations are used. Supporting evidence is supplied when quantifying the performance of the proposed MASC in a real-time context parameterised by T_{max}. An upper bound of the 'speedup' offered by MAS is $E1=2^K$ for a stationary synthesis application mapped into the terminal subbands of the subband hierarchy. Filterbank computation, logical deadband exclusion and note non-stationarity reduce speedup, but a property of MAS is that computation is optimised to the note dynamics of the application in question and the generality of classical AS is preserved. Benchmarking is an area worthy of further investigation.

A survey of the two schools of sinusoidal oscillator design concludes that a phase-accumulator form is preferable to a recursive form for MASC implementation because of (i) the ease of dynamic frequency control and (ii) only one state variable is required in place of two for a 2nd order recursive system. The commonly perceived bottleneck of sine transformation in a TOB via large LUTs can be solved using a pipeline form of linear-interpolated LUT where the interpolation is executed by a dedicated multiplier with an optimised wordlength. Another method is the CORDIC algorithm which can generate complex sinusoids in association with the PEF filterbank. Modifications are

documented which result in the specification of an oscillator core based upon a CORDIC pipeline: a simulation shows how pipeline length and wordlength affect the output S/N. However, it is concluded that a pipelined linear-interpolated LUT provides the best solution for VLSI implementation.

The specification and design of the proposed VLSI MASC starts from a systems analysis of the processes in a hypothetical MAS synthesiser which form a synchronous logical pipeline. Data bandwidth proliferates in the MOB and is mapped into the MASC for computational acceleration; all other processes are mapped into software. The MASC is allied to a Shared Memory (SM) which is mapped via a bus interface into the Host CPU's address space to facilitate the required complexity of IPC. The Host has highest priority SM access and blocks the MASC, but as Host SM access is relatively infrequent compared to saturated MASC-SM traffic, MASC performance degradation is minimal. MASC functionality is encapsulated in the prototype Oscillator Descriptor which, in the context of an economical single data-bus MASC and coarse time-quantisation of PWL breakpoints, leads to the concept of concurrent MOB burst operation with interleaving of MASC-SM OD traffic. Dataflow in the MASC-SM system is thus organised on the principle of a *memory hierarchy* to optimise throughput.

Multirate scheduling and Host-MASC synchronisation are facilitated by a frame timing structure of length T_{frame} in which interleaved burst MOB operation is optimised for the lowest subbands at level K. Higher subband sample rates are generated by extending MOB bursts. Organising the round-robin schedule of OD's as a linked list facilitates (i) efficient ordering of OD's by subband and filterbank to enable, respectively, burst accumulation and mutually exclusive SM access to MOB-FBP buffers in the proposed synchronisation algorithm and (ii) efficient resource allocation by the Host when large blocks of OD's are manipulated. The logical process pipeline is driven by the FBP with which the MASC obeys a 'busy-waiting' protocol. Within frames, the Host and MASC operate with relative asynchronicity to enable the throughput of each to be maximised without the complexities of synchronous IPC in the context of a dynamically configured OD schedule. Finally, a dataflow diagram and the logic of the major functional units within the MASC are presented.

A fundamental fact to emerge from a parameterised analysis of MASC performance is that max(S) is a function of T_{max} and the MASC-SM databus bandwidth: to this extent the MASC is input / output bound and optimisation by MAS does not figure. The influence of MAS is to reduce the required internal MASC clock frequency from that required by a classical AS form of MASC by a factor of 2^{-K} which can be exploited by using cheaper, slower ASIC technologies: for $K=3$, internal MASC clock frequency is reduced eightfold. Assuming allocation to the deepest subbands at level K in a subband hierarchy, the same max(S) is achievable as in a classical AS form of MASC. However, a complexity increase arises in the MASC from supporting multirate operation by extended bursts due to the requirement for a variable length accumulation FIFO. Additionally, a software overhead is imposed by filterbank computation.

Future work can adopt a number of directions. Further refinements to the details of QMF subband hierarchy implementation are possible e.g. a deeper analysis of optimal K in the context of comprehensive benchmarking. Also, the development of a MASC behavioural simulation and prototype will permit a more detailed analysis of actual real-time performance than that anticipated at the level of a functional specification. Another area which is unexplored is the use of MAS as a software synthesis technique. However, a final comment upon this thesis is that the proposed MAS algorithm and consequent MASC architecture are, in the opinion of the author, about as far as TOB-based AS implementations can be realistically optimised: a significant improvement over the classical TOB is feasible. So this work may be comprehended as both (i) a necessary summary to a line of research that has fallen out of favour before achieving its natural level of maturity and (ii) a potential starting point for a revival of interest in TOB methods.

Appendix A: SHARC Database Structure

alto_trombone	CB_pizz	piccolo
altoflute_vibrato	cello_martele	trombone
Bach_trumpet	cello_muted_vibrato	trombone_muted
bass_clarinet	cello_pizzicato	tuba
bass_trombone	cello_vibrato	viola_martele
bassflute_vibrato	contrabass_clarinet	viola_muted_vibrato
bassoon	contrabassoon	viola_pizzicato
Bb_clarinet	Eb_clarinet	viola_vibrato
C_trumpet	English_horn	violin_martele
C_trumpet_muted	flute_vibrato	violin_muted_vibrato
CB	French_horn	violin_pizzicato
CB_martele	French_horn_muted	violin_vibrato
CB_muted	oboe	violinensemb

Appendix B: Orchestration of Enigma Variations

Full Score Section	SHARC Timbre	Number of Voices	Filterbank#
Bassoon 1	bassoon	1	1
Bassoon 2	bassoon	1	1
Contrabassoon	contrabassoon	1	1
Cello	cello_martele	9	2
Clarinet 1	Bb_clarinet	1	1
Clarinet 2	Bb_clarinet	1	1
Double Bass	CB	6	3
Flute1	flute_vibrato	1	1
Flute 2	flute_vibrato	1	1
Horns 1	French_horn	1	4
Horns 2	French_horn	1	4
Horns 3	French_horn	1	4
Horns 4	French_horn	1	4
Oboe 1	oboe	1	1
Oboe 2	oboe	1	1
Timpani	N/A		
Trombone 1	trombone	1	5
Trombone 2	trombone	1	5
Trombone 3	tuba	1	5
Trombone 4	bass_trombone	1	5
Trumpet 1	C_trumpet	1	5
Trumpet 2	C_trumpet	1	5
Trumpet 3	C_trumpet	1	5
Violas	viola_martele	9	6
Violins 1	violin_martele	12	7
Violins 2	violin_martele	12	8

References

Abuelhaija A. I. and Alibrahim M. M. (1986), "Improving Performance of Digital Sinusoidal Oscillators by Means of Error Feedback Circuits", *IEEE Trans. on Circuits and Systems*, vol. 33, no. 4, pp. 373-380.

Adams R. and Kwan T. (1994), "A Stereo Asynchronous Sample-Rate Converter for Digital Audio", *IEE Journal of Solid-State Circuits*, vol. 29, no. 4, pp. 481-488.

Ahmed H. M. (1985), "Alternative Arithmetic Unit Architectures for VLSI Digital Signal Processing", *in VLSI and Modern Signal Processing, Edited by S.Y. Kung, H.J. Whitehouse and T. Kailath*, Englewood Cliffs, NJ, USA, Prentice-Hall, pp. 277-303.

Brown J. C. and Puckette M. S. (1992), "An Efficient Algorithm for the Calculation of the Constant-Q Transform", *Journal of the Acoustical Society of America*, vol. 92, no. 5, pp. 2698-2701.

Charbonneau G. R. (1981), "Timbre and the Perceptual Effects of Three Types of Data Reduction", *Computer Music Journal*, vol. 5, no. 2, pp. 10-19.

Chowning J. M. (1973), "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation", *Journal of the Audio Engineering Society*, vol. 21, no. 7, pp. 526-534.

Comerford P. (1993), "Simulating an Organ with Additive Synthesis", *Computer Music Journal*, vol. 17, no. 2, pp. 55-65.

De Poli G. (1983), "A Tutorial on Digital Sound Synthesis Techniques", *Computer Music Journal*, vol. 7, no. 4, reprinted in *The Music Machine - Selected Readings from Computer Music Journal, Edited by C. Roads*, MIT Press, 1989, pp. 429-447.

Eaglestone B. and Oates S. (1990), "Analytical Tools for Group Additive Synthesis", *Int. Computer Music Conf.*, Univ. of Glasgow, UK, pp. 66-68.

Elgar E. (1899), "Enigma: Variations on an Original Theme", London, Novello & Company.

Feiten B. H. and Ungvary T. (1990), "Sound Database Using Spectral Analysis Reduction and an Additive Synthesis Model", *Int. Computer Music Conf.*, Univ. of Glasgow, UK, pp. 72-74.

Fitz K. and Haken L. (1995), "Bandwidth Enhanced Sinusoidal Modeling in Lemur", *Proc. of Int. Computer Music Conf.*, Banff, Canada, pp. 154-157.

Fitz K. Walker W. and Haken L. (1992), "Extending the McAulay-Quatieri Analysis for Synthesis with a Limited Number of Oscillators", *Proc. of Int. Computer Music Conf.*, San Jose State University, CA, USA, pp. 381-382.

Flanagan M. J. and Zimmerman G. A. (1993), "Spur-Reduced Digital Sinusoid Generation Using Higher-Order Phase Dithering", *Proc. of 27th Asilomar Conf. of Signals, Systems and Computers*, Pacific Grove, CA, USA, pp. 826-830.

Fliege N. J. and Wintermantel J. (1992), "Complex Digital Oscillators and FSK Modulators", *IEEE Trans. on Signal Processing*, vol. 40, no. 2, pp. 333-342.

Freed A. (1995), "Bring your Own Control to Additive Synthesis", *Proc. of Int. Computer Music Conf.*, Banff, Canada, pp. 303-306.

Freed, A. Rodet X. and Depalle P. (1993), "Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware", *Proc. of Int. Computer Music Conf.*, Waseda University, Tokyo, pp. 98-101.

Gabor D. (1947), "Acoustical Quanta and the Theory of Hearing", *Nature*, vol. 159, no. 4044, pp. 591-592.

George E. B. and Smith M. J. T. (1992), "Analysis-by-Synthesis / Overlap-Add Sinusoidal Modeling Applied to the Analysis and Synthesis of Musical Tones", *Journal of the Audio Engineering Society*, vol. 40, no. 6, pp. 497-516.

Goodwin M. (1996), "Residual Modelling in Music Analysis-Synthesis", *Proc. of Int. Conf. on Acoustics Speech and Signal Processing*, Atlanta.

Goodwin M. and Kogon A. (1995), "Overlap-Add Synthesis of Nonstationary Sinusoids", *Proc. of Int. Computer Music Conf.*, Banff, Canada, pp. 355-356.

Goodwin M. and Rodet X. (1994), "Efficient Fourier Synthesis of Nonstationary Sinusoids", *Proc. Int. Computer Music Conf.*, Aarhus, Denmark, pp. 333-334.

Gordon J. W. and Smith J. O. (1985), "A Sine Generation Algorithm for VLSI Applications", *Proc. of Int. Computer Music Conf.*, Simon Fraser University, Burnaby, British Columbia, Canada, pp. 165-168.

Grey J. and Moorer J. (1977), "Perceptual Evaluation of Synthesized Musical Instrument Tones", *Journal of the Acoustical Society of America*, vol. 62, pp. 454-462.

Haken L. (1991), "Computational Methods for Real-Time Fourier Synthesis", *IEEE Trans. on Signal Processing*, vol. 40, no. 2, pp. 2327-2329.

Hart, J. Naylor P. and Tanrikulu O. (1993), "Polyphase Allpass Structures for Sub-band Acoustic Echo Cancellation", *Proc. of EUROSPEECH'93*, Berlin, vol. 3, pp. 1813-1816.

Hill R. D. (1991), "The Cro-Magnon Advanced Additive Analysis / Synthesis System", *Proc. of Int. Computer Music Conf.*, *Proc. of Int. Computer Music Conf.*, McGill University, Montreal, pp. 169-176.

Holm F. (1994), "Control of Frequency and Decay in Oscillating Filters Using Multirate Techniques", *Proc. Int. Computer Music Conf.*, Aarhus, Denmark, pp. 372-375.

Horner A. and Cheung N. (1995), "Genetic Algorithm Optimisation of Additive Synthesis Envelope Breakpoints and Group Synthesis Parameters", *Proc. of Int. Computer Music Conf.*, Banff, Canada, pp. 215-222.

Horner A. Beauchamp J. and Haken L. (1992), "Wavetable and FM Matching Synthesis of Musical Instrument Tones", *Proc. of Int. Computer Music Conf.*, San Jose State University, CA, USA, pp. 18-21.

Horner A. Beauchamp J. and Haken L. (1993), "Methods for Multiple Wavetable Synthesis of Musical Instrument Tones", *Journal of the Audio Engineering Society*, vol. 41, no. 5, pp. 336-356.

Houghton A. D. Fisher A. J. and Malet T. F. (1995), "An ASIC for Digital Additive Sine-wave Synthesis", *Computer Music Journal*, vol. 19, no. 3, pp. 26-29.

Hu Y. H. (1992a), "CORDIC-Based VLSI Architectures for Digital Signal Processing", *IEEE Signal Processing Magazine*, July 1992, pp. 16-35.

Hu Y. H. (1992b), "The Quantization Effects of the CORDIC Algorithm", *IEEE Trans. on Signal Processing*, vol. 40, no. 4, pp. 16-35.

Jaffe D. A. (1995), "Ten Criteria for Evaluating Digital Synthesis Techniques", *Computer Music Journal*, vol. 19, no. 1, pp. 76-87.

Jansen C. (1991), "Sine Circuitu, 10,000 High Quality Sine Waves Without Detours", *Proc. of Int. Computer Music Conf.*, McGill University, Montreal, pp. 222-225.

Kahrs M. (1981), "Notes on Very-Large Scale Integration and the Design of Real-Time Digital Sound Processors", *Computer Music Journal*, vol. 5, no. 2, reprinted in *The Music Machine - Selected Readings from Computer Music Journal, Edited by C. Roads*, MIT Press, 1989, pp. 623-631.

Kaper H., Ralley D., Restrepo J. and Tipei S. (1995), "Additive Synthesis with DIASS_M4C on Argonne National Laboratory's IBM POWERparallel System (SP)", *Proc. of Int. Computer Music Conf.*, Banff, Canada, pp. 351-352.

Kaplan S. J. (1981), "Developing a Commercial Digital Synthesizer", *Computer Music Journal, vol. 5, no. 3, reprinted in The Music Machine - Selected Readings from Computer Music Journal, Edited by C. Roads*, MIT Press, 1989, pp. 611-622.

Karplus K. and Strong A. (1983), "Digital Synthesis of Plucked-String and Drum Timbres", *Computer Music Journal*, vol. 7, no. 2, pp. 43-55.

Kleczkowski P. (1989), "Group Additive Synthesis", *Computer Music Journal*, vol. 13, no. 1, pp. 12-20.

Lee E. A. and Messerchmitt D. G. (1987a), "Pipeline Interleaved Programmable DSP's: Architecture", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 35, no. 9, pp. 1320-1333.

Lee E. A. and Messerchmitt D. G. (1987b), "Pipeline Interleaved Programmable DSP's: Synchronous Dataflow Programming", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 35, no. 9, pp. 1334-1345.

Levine S. N. (1996a), "Critically Sampled Third Octave Filter Banks", *Proc. Int. Computer Music Conf.*, HKUST, Hong Kong, pp. 301-304.

Levine S. N. (1996b), "Effects Processing on Audio Subband Data", *Proc. Int. Computer Music Conf.*, HKUST, Hong Kong, pp. 328-331.

Limberis A. and Bryan J. (1993), "An Architecture for a Multiple Digital Signal Processor Based Music Synthesizer with Dynamic Voice Allocation", *Audio Engineering Society 95th Convention*, New York, (preprint).

Marks M. A. (1988), "Resource Allocation in an Additive Synthesis System for Audio Waveform Generation", *Proc. of Int. Computer Music Conf.*, Cologne, pp. 378-382.

Mauchly J. W. and Charpentier A. J. (1987), "Practical Considerations in the Design of Music Systems using VLSI", *Proc. of 5th Int. Audio Engineering Society Conf.*, Los Angeles, pp. 28-36.

Masri P. and Bateman A. (1996), "Improved Modelling of Attack Transients in Music Analysis-Resynthesis", *Proc. Int. Computer Music Conf.*, HKUST, Hong Kong, pp. 100-103.

McAulay R. J. and Quatieri T. F. (1986), "Speech Analysis / Synthesis Based on a Sinusoidal Representation", *IEEE Trans. on Acoustics Speech and Signal Processing*, vol. 34, no. 4, pp. 744-754.

McAulay R. J. and Quatieri T. F. (1987), "Multirate Sinusoidal Transform Coding at Rates From 2.4 Kbps to 8 Kbps", *Proc. of Int. Conf. on Acoustics Speech and Signal Processing*, Dallas, TX, USA, pp. 1645-1648.

McAulay R. J. and Quatieri T. F. (1988), "Computationally Efficient Sine-Wave Synthesis and its Application to Sinsuoidal Transform Coding", *Proc. of IEEE Conf. on Acoustics Speech and Signal Processing*, pp. 370-373.

Mehrgardt S. (1983), "Noise Spectra of Digital Sine-Generators Using the Table-Lookup Method", *IEEE Trans. on Acoustics Speech and Signal Processing*, vol. 31, no. 4, pp. 1037-1039.

Meyer J. M. (1993), "The Sound of the Orchestra", *Journal of the Audio Engineering Society*, vol. 41, no. 4, pp. 203-213.

Moore F. R. (1977), "Table Lookup Noise for Sinusoidal Digital Oscillators", *Computer Music Journal*, vol. 1, no. 2, pp. 26-29.

Moorer J. A. (1982), "The Lucasfilm Audio Signal Processor", *Computer Music Journal*, vol. 6, no. 3, reprinted in *The Music Machine - Selected Readings from Computer Music Journal, Edited by C. Roads*, MIT Press, 1989, pp. 599-609.

Moorer J.A. (1976), "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulas", *Journal of the Audio Engineering Society*, vol. 24, no. 9, pp. 717-727.

Moorer J.A. (1977), "Signal Processing Aspects of Computer Music: A Survey", *Computer Music Journal*, vol. 1, no. 1, pp. 4-37.

Moorer J.A., Grey J. M. and Strawn J. (1978), "Lexicon of Analysed Tones. Part3: The Trumpet", *Computer Music Journal*, vol. 2, no. 2, pp. 23-31.

Nuttall A. H. (1981), "Some Windows with Very Good Sidelobe Behavior", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 29, no. 1, pp. 84-91.

Osaka N. (1995), "Timbre Interpolation of Sounds Using a Sinusoidal Model", *Proc. of Int. Computer Music Conf.*, Banff, Canada, pp. 408-411.

Pan D. (1994), "An Overview of the MPEG Audio Compression Algorithm", *Conference on Digital Video Compression on Personal Computers: Algorithms and Technologies*, San Jose, CA, USA, vol. 2187, ch. 33, pp. 260-273.

Parks T. H. and McClellan J. H. (1972), "Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase", *IEEE Trans. on Circuit Theory*, vol. 19, no. 2, pp. 189-194.

Phillips D. K., Purvis A. and Johnson S. (1996a), "Multirate Additive Synthesis", *Proc. Int. Computer Music Conf.*, HKUST, Hong Kong, pp. 496-499.

Phillips D. K., Purvis A. and Johnson S. (1996b), "On An Efficient VLSI Architecture for the Multirate Additive Synthesis of Musical Tones", submitted for publication in the *Euromicro Journal of Systems Architecture*.

Phillips D., Purvis A. and Johnson S. (1994), "A Multirate Optimisation for Real-Time Additive Synthesis", *Proc. Int. Computer Music Conf.*, Aarhus, Denmark, pp. 364-367.

Rioul O. and Vetterli M. (1991), "Wavelets and Signal Processing", *IEEE Signal Processing Magazine*, October 1991, pp. 14-38.

Risset J. and M. Mathews (1969), "Analysis of Musical Instrument Tones", *Physics Today*, vol. 22, no. 2, pp. 23-30.

Risset. J. (1965), "Computer Study of Trumpet Tones", *Journal of the Acoustic Society of America (Abstracts)*, vol. 38, p. 912.

Roads C. (1978), "Automated Granular Synthesis of Sound", *Computer Music Journal*, vol. 2, no. 2, pp. 61-62.

Rodet X. (1984), "Time Domain Formant-Wave-Function Synthesis", *Computer Music Journal*, vol. 8, no. 3, pp. 9-14.

Rodet X. (1992), "Nonlinear Oscillator Models of Musical Instrument Excitation", *Proc. of the Int. Computer Music Conf.*, San Jose State University, CA, USA, pp. 412-413.

Rodet X. and Depalle P. (1992). "A New Additive Synthesis Method using Inverse Fast Fourier Transform and Spectral Envelopes", *Proc. of the Int. Computer Music Conf.*, San Jose State University, CA, USA, pp 410-411.

Sandell G. J. (1991), "A Library of Orchestral Instrument Spectra", *Proc. of Int. Computer Music Conf.*, McGill University, Montreal, pp. 98-101.

Sandell G. J. (1994), *SHARC Timbre Database Release 0.90 ("beta") November 1994*, Published at University of Sussex Website,.

Sandler M. (1989), "High Complexity Resonator Structures for Formant Synthesis of Musical Instruments", *15th Euromicro Symposium on Microprocessing and Microprogramming*, Cologne, pp. 557-562.

Sasaki L. H. and Smith K. C. (1980), "A Simple Data Reduction Scheme for Additive Synthesis", *Computer Music Journal*, vol. 4, no. 1, pp. 22-24.

Schindler K. W. (1984), "Dynamic Timbre Control for Real-Time Digital Synthesis", *Computer Music Journal*, vol. 8, no. 1, 1984, pp. 28-42.

Serra M. H. Rubine D. and Dannenberg R. (1990), "Analysis and Synthesis of Tones by Spectral Interpolation", *Journal of the Audio Engineering Society*, vol. 38, no. 3, pp. 111-128.

Serra X. (1994), "Sound hybridization based on a deterministic plus stochastic decomposition model", *Proc. Int. Computer Music Conf.*, Aarhus, Denmark, pp. 348-351.

Serra X. and Smith J. O. (1990), "Spectral Modelling Synthesis: A Sound Analysis / Synthesis System Based on a Deterministic plus Stochastic Decomposition", *Computer Music Journal*, vol. 14, no. 4, pp. 12-24.

Smith J. O. (1987), "Music Applications of Digital Waveguides", CCRMA Technical Report STAN-M-39, CCRMA, Stanford University, CA, USA.

Smith J. O. (1991), "Viewpoints on the History of Digital Synthesis", Keynote Paper, *Proc. of Int. Computer Music Conf.*, McGill University, Montreal, pp. 1-10.

Smith J. O. and Cook P. R. (1992), "The Second-Order Digital Waveguide Oscillator", *Proc. of Int. Computer Music Conf.*, San Jose State University, CA, USA, pp. 150-153.

Snell J. (1977), "Design of a Digital Oscillator That Will Generate up to 256 Low-Distortion Sine Waves in Real-Time", *Computer Music Journal*, vol. 1, no. 2, pp. 4-25.

Stapleton J. C. and Bass S. C. (1988), "Synthesis of Musical Tones Based on the Karhunen-Loève Transform", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 36, no. 3, pp. 305-319.

Stilson T. and Smith J. O. (1996), "Alias-Free Digital Synthesis of Classic Analog Waveforms", *Proc. Int. Computer Music Conf.*, HKUST, Hong Kong, pp. 332-335.

Strawn J. (1980), "Approximation and Syntactic Analysis of Amplitude and Frequency Functions for Digital Sound Synthesis", *Computer Music Journal*, vol. 4, no. 3,

reprinted in *The Music Machine - Selected Readings from Computer Music Journal*, *Edited by C. Roads*, MIT Press, 1989, pp. 671-692.

Strawn J. (1987), "Analysis of Musical Transitions Using the Discrete Short-time Fourier Transform", *Journal of the Audio Engineering* Society, vol. 35, no. 1/2, pp. 3-13.

Tierney J., Rader C. M. and Gold B. (1971), "A Digital Frequency Synthesizer", *IEEE Trans. on Audio and Electroacoustics*, vol. 19, 1971, pp. 48-57.

Van Duyne S. A. and Smith J. O. (1994), "A Simplified Approach to Modeling Dispersion Caused by Stiffness in Strings and Plates", *Proc. Int. Computer Music Conf.*, Aarhus, Denmark, pp. 407-410.

Van Duyne S. A., Pierce J. R. and Smith J. O. (1994), "Traveling Wave Implementation of A Lossless Mode-Coupling Filter and The Wave Digital Hammer", *Proc. Int. Computer Music Conf.*, Aarhus, Denmark, pp. 411-418.

Wawrzynek J. C. and Mead C. A. (1985), "A VLSI Architecture for Sound Synthesis", in *VLSI Signal Processing: A Bit-Serial Approach, Edited by Denyer & Renshaw*, Reading, MA, USA, Addison Wesley, pp 277-297.

Wawrzynek J. C. and von Eicken T. (1990), "VLSI Parallel Processing for Musical Sound Synthesis", *Int. Computer Music Conf.*, Univ. of Glasgow, UK, pp. 136-139.

Wessel D. L. (1979), "Timbre Space as a Musical Control Structure", *Computer Music Journal*, vol. 3, no. 2, pp. 45-52.

White P. (1995), "Oberheim FAR Resynthesis Technology", *Sound on Sound Magazine*, January 1995, p. 22.

Bibliography

Burns A. and Wellings A. (1989), *Real-Time Systems and Their Programming Languages*, Addison-Wesley, Wokingham, UK.

Chamberlin H. (1980), *Musical Applications of Microprocessors*, 1st Edition, Hayden Books, NJ, USA.

Crochiere R. E. and Rabiner L. R. (1983), *Multirate Digital Signal Processing*, Englewood Cliffs, NJ, USA, Prentice-Hall.

DeCegama A. L. (1989), *The Technology of Parallel Processing: Parallel Processing Architectures and VLSI Hardware Volume 1*, Englewood Cliffs, NJ, USA, Prentice-Hall.

Helmholtz H. L. F. V. (1863), *On the Sensations of Tone as a Physiological Basis for the Theory of Music*, 1863, (1954 transl. of 1877 edition), Dover, New York.

Hennesy J. L. and Patterson D. A. (1990), *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Mateo, CA, USA.

Higgins R. J. (1990), *Digital Signal Processing in VLSI*, Englewood Cliffs, NJ, USA, Prentice-Hall, pp. 524-555.

Holst I. (1963), *An ABC of Music*, Reprinted 1988, The Chaucer Press, Bungay, Suffolk, UK.

Leiss E. L. (1995), *Parallel and Vector Computing: A Practical Introduction*, McGraw-Hill, New York.

Lister A. M. and Eager R. D. (1988), *Fundamentals of Operating Systems*, 4th Edition, Macmillan Education, Basingstoke, UK.

Manning P. (1985), *Electronic and Computer Music*, Clarendon Press, Oxford, UK.

Moore F. R. (1990), *Elements of Computer Music*, Englewood Cliffs, NJ, USA, Prentice Hall.

Oppenheim A V. and Schafer R. W. (1989), *Discrete-Time Signal Processing*, Englewood Cliffs, NJ, USA, Prentice Hall.

Oppenheim A. V., Willsky A. S. and Young I .T. (1983), *Signals and Systems*, London, Prentice-Hall International.

Parsons T. (1987), *Voice and Speech Processing*, McGraw-Hill, New York.

Plomp R. (1976), *Aspects of Tone Sensation*, New York, Academic Press.

Rabiner L. R. and Gold B. (1975), *Theory and Applications of Digital Signal Processing*, Englewood Cliffs, NJ, USA, Prentice Hall.

Roads C. (Ed.) 1989, *The Music Machine - Selected Readings from Computer Music Journal*, MIT Press.

Roads C. and Strawn J. (Ed.) 1985, *The Foundations of Computer Music,* The MIT Press.

Taylor C.A. (1965), *The Physics of Musical Sounds*, The English Universities Press, London.

Tenenbaum A. M. and Augenstein M. J. (1986), *Data Structures Using Pascal*, 2nd Edition, London, Prentice-Hall International.

Vaidyanathan P. P. (1993), *Multirate Systems and Filter Banks*, Englewood Cliffs, NJ, Prentice-Hall.

Wakerly J. F. (1990), *Digital Design Principles and Practices*, London, Prentice-Hall International.

Weltner K. and Grosjean J., Schuster P., Weber W. J. (1986), *Mathematics for Engineers and Scientists*, Stanley Thornes Publishers, Cheltenham, UK.

Weste N. H. E. and Eshraghian K. (1988), *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, Reading, MA, USA.

Westrup J. and Harrison F. L. (1959), *Collins Encyclopaedia of Music*, Chancellor Press, London, 2nd Edition 1984.

Glossary of Terms

Abbreviations

AM   Amplitude Modulation

AS   Additive Synthesis

ASIC   Application-Specific Integrated Circuit

CD   Compact Disk (16-bit stereo audio at f_s=44.1kHz)

CMOS   Complimentary Metal Oxide Semiconductor

CORDIC   COordinate Rotation DIgitial Computer

CPU   Central Processing Unit

DFT   Discrete Fourier Transform

EC   Envelope Cache

FBP   Filterbank Processor

FFT   Fast Fourier Transform

FFT^{-1} IFFT AS Synthesis Technique proposed by (Rodet and Depalle, 1992)

FIFO   First-In First-Out Buffer

FIR   Finite Impulse Response

FM   Frequency Modulation

IDFT   Inverse Discrete Fourier Transform

IFFT   Inverse Fast Fourier Transform

IFT   Inverse Fourier Transform

IIR   Infinite Impulse Response

IPC   Inter-Processor Communication

LUT   Look-Up Table

MAS	Multirate Additive Synthesis
MASC	Multirate Additive Synthesis Coprocessor
MOB	Multirate Oscillator Bank
NMOS	N-channel Metal Oxide Semiconductor
OD	Oscillator Descriptor
OLA	Overlap-Add
PA-IIR	Polyphase Allpass IIR QMF stage design
PC	Personal Computer
PEF	Physical Exclusion (of Δ_f) Filterbank
PM-FIR	Parks-McClellan equiripple FIR QMF stage design
PWL	Piecewise Linear
Q	As documented in section 3.1.4
QMF	Quadrature Mirror Filterbank
RMS	Root Mean Square
ROM	Read Only Memory
S/N	Signal to Noise Ratio
SHARC	Sandell Harmonic Archive (Sandell, 1994)
SM	Shared Memory
SMS	Spectral Modelling Synthesis (Serra and Smith, 1990)
SSB	Single Side Band
STFT	Short-Time Fourier Transform
TOB	Traditional Oscillator Bank as defined in section 1.3.2
VLSI	Very Large Scale Integration

Commonly Used Mathematical Symbols

S	Number of oscillators in an AS application
$A_i[n]$	Time series of amplitude envelope values for i^{th} oscillator of S oscillators
$F_i[n]$	Time series of frequency envelope values for i^{th} oscillator of S oscillators
$\phi_i[n]$	Time series of phase envelope values for i^{th} oscillator of S oscillators
$\Phi_i[n]$	Time series of phase accumulator values for i^{th} oscillator of S oscillators
$\Omega_i[n]$	Subband normalised form of $F_i[n]$
f_s	An industry-standard digital audio sampling rate e.g. CD at 44.1kHz
$f_{opt}(x)$	The optimum time-invariant sampling rate for alias-free synthesis of x
$f_{max}(x)$	The upper frequency bound of of x during its lifecycle
$f_{min}(x)$	The lower frequency bound of of x during its lifecycle
x	An arbitrary sinusoidal oscillator allocated to a partial in note-based AS
T_{max}	Latency tolerance for real-time synthesis
I	Interpolation factor
D	Decimation factor
δ_p	Filter passband ripple
δ_s	Filter stopband ripple
Δ_f	Filter transition width
K	Number of subbands, Depth of QMF subband hierarchy
$s_{k,l}$	Subband set of subband hierarchy
k	Depth in subband hierarchy ($0 \leq k \leq K$)
l	Integer-subband index at level k in subband hierarchy ($1 \leq l \leq 2^k$)
$T_{fb}(k)$	QMF filterbank latency from level k
λ	PEF filterbank oversampling factor

M	Latency of an FIR QMF stage
u_{mas}	Unit cost of an oscillator from a MOB
u_{as}	Unit cost of an oscillator from a TOB
$v(K)$	Unit cost of a filterbank (as a function of K)
n_k	Number of oscillators allocated to subband k
ε	Ratio of included to excluded partial power
τ	Pitch modulation range ($\pm\tau$) required by a note in semitones
$E1$	Benchmarked AS / MAS control bandwidth requirements ratio
$E2$	Benchmarked AS / MAS multiplication requirements ratio
$f_s(x)$	Sample rate of oscillator x
N_{burst}	Length of standard (non-extended) MOB burst
T_{frame}	Duration of a MASC frame
T_{part}	Duration of a filterbank partition in MASC frame
T_{tot}	Total latency of a MAS synthesiser
$T_{fb}(K)$	Maximum latency of a QMF-style filterbank i.e. that from level K