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

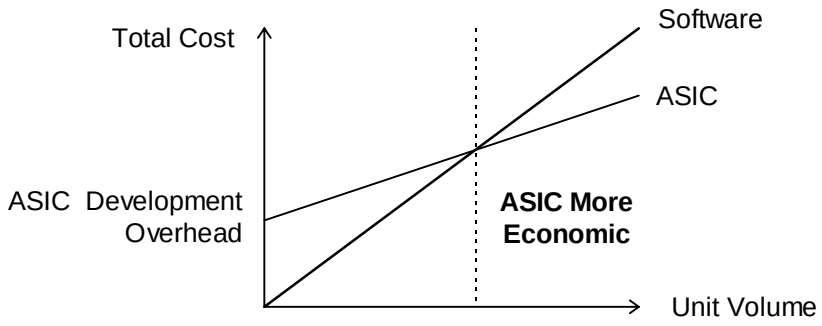

Figure 8.1 Comparison of the Economics of Software and ASIC Implementations

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.

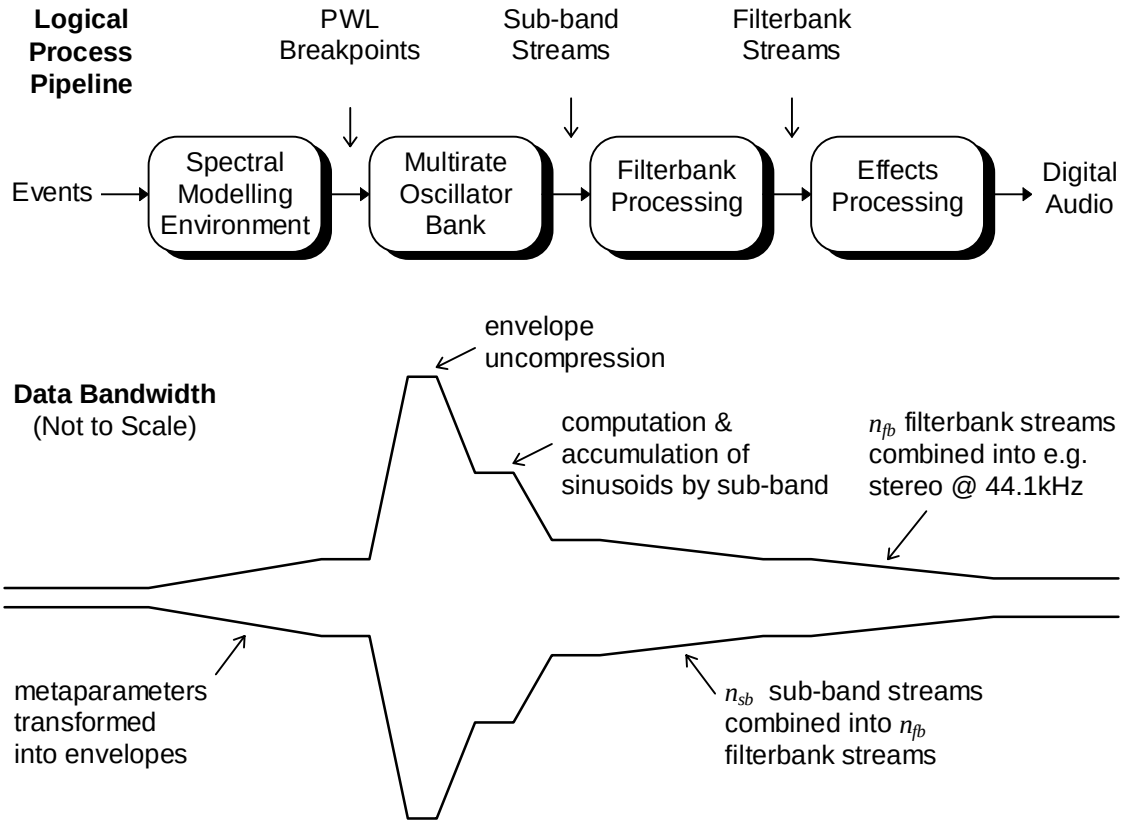

Figure 8.2 Processes and Control Data Bandwidth in a MAS Synthesiser

In the MOB, $A_i[n]$ and $F_i[n]$ envelopes for $1 \le i \le S$ 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, $n_{sb}<<S$. 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

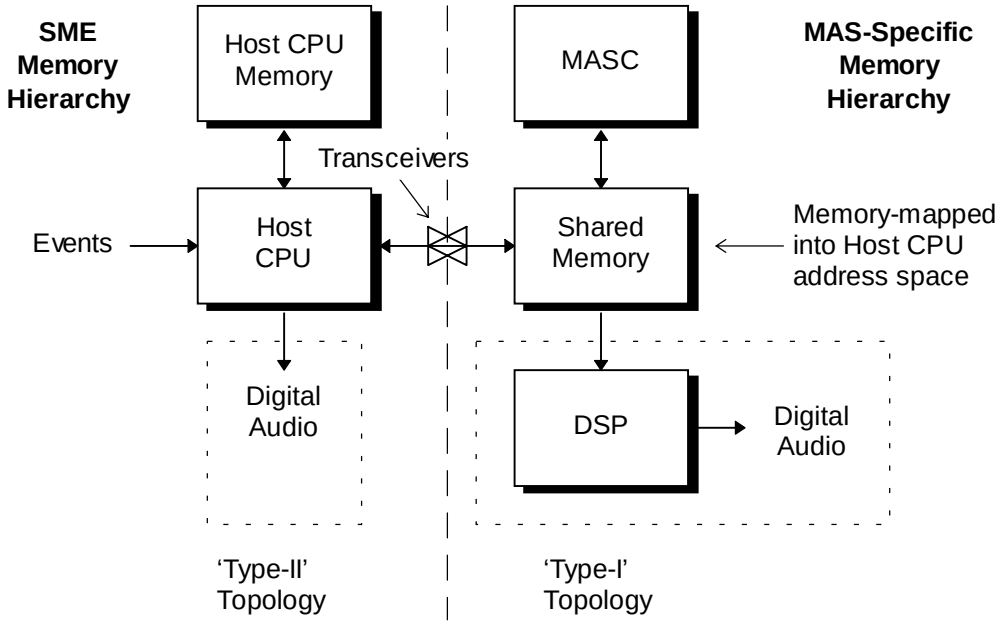

Figure 8.3 Architectural Forms of Multiprocessor MAS Synthesiser

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

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.

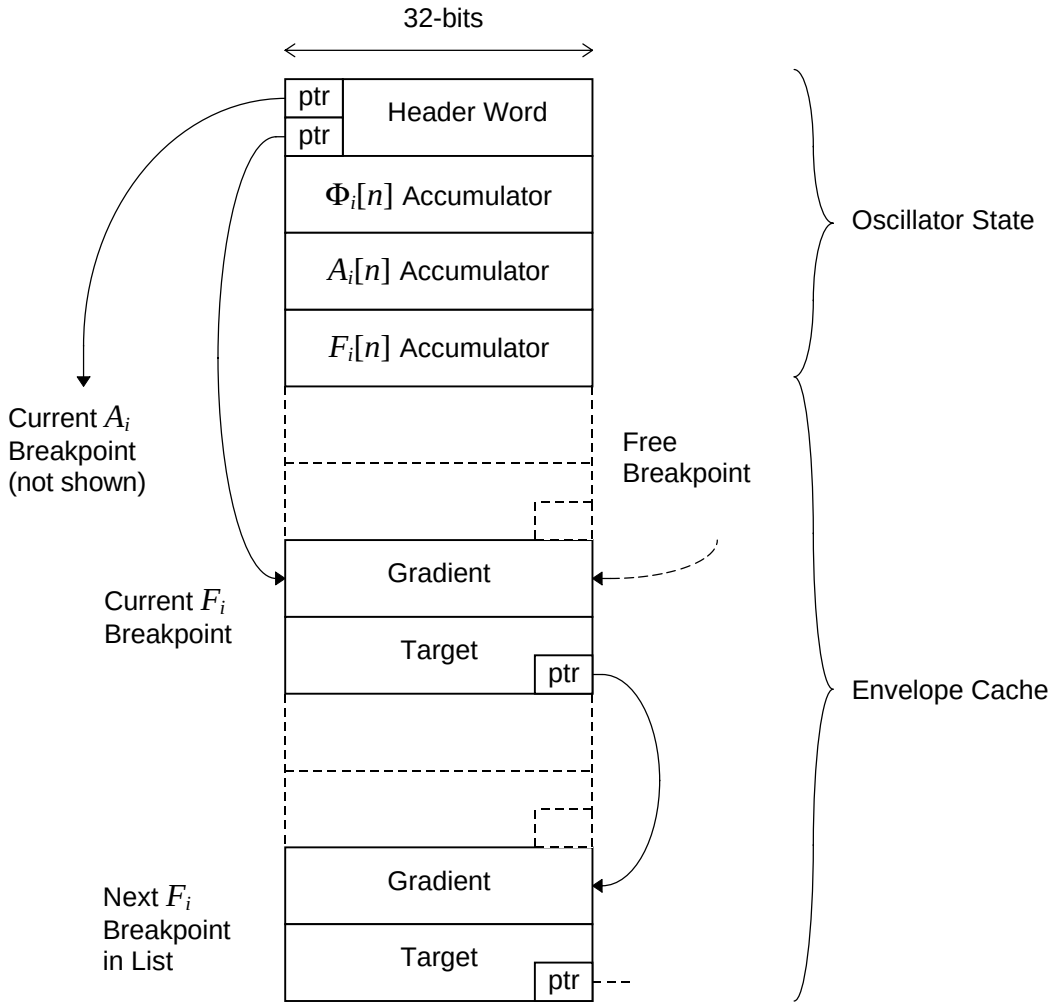

Figure 8.4 Oscillator Descriptor Organisation

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

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.

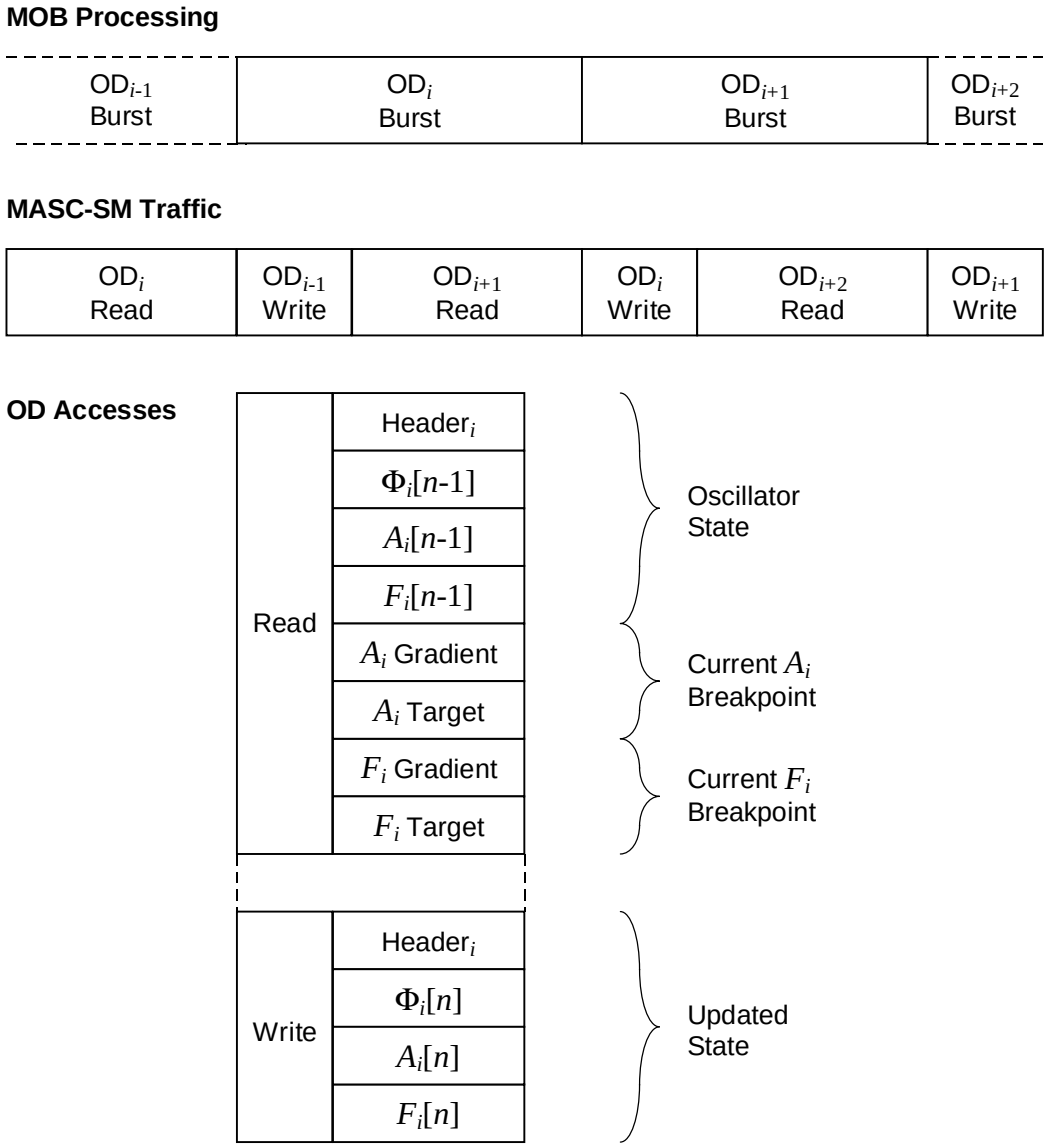

Figure 8.5 Interleaved Burst Processing

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.

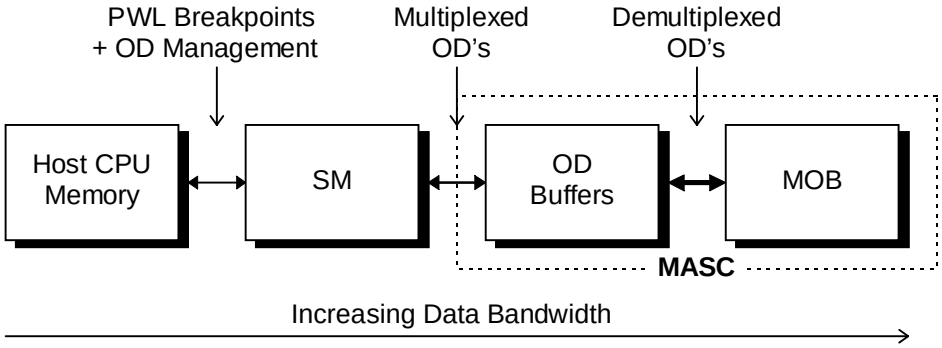

Figure 8.6 Model of MAS Memory Hierachy