

Fig.1 Software package for the macrocell system

An alternative to all of these is the *silicon compiler* approach [2,3]. The design is specified procedurally, usually by providing a text file as input to the compiler. The compiler then generates a mask-level description of the circuit layout. This paper describes a compiler developed specifically for the purpose of generating signal processing IC's. Target applications are speech processing (vocoders, noise canceling, speech recognition), telecommunications (modems, line equalizers, echo cancelers), and digital audio (equalization, special effects).

An important aspect of the system is the use of *macrocells*. A macrocell is a large block of circuitry assembled by tiling (arraying) cells from a cell library in two dimensions. This approach is very flexible, allowing the size and function of each macrocell to be customized for the particular design.

The complete software system is shown in Fig.1. All software is written in the C programming language, and runs under the 4.2 BSD version of the UNIX operating system.

The input to the system is the *design file*. A design file for a typical application consists of a few pages of text. Because the silicon compiler generates multiprocessor IC's, the design file contains hardware options and symbolic microcode for each processor in the design. In addition, there is a section which describes interprocessor and off-chip communications.

The emulator enables the designer to debug and verify his design file input. In an interactive mode, the emulator has a full set of debugging commands: breakpoints, tracing, single-step, setting and displaying variables. A batch mode allows non-real-time simulation of the signal processor design. This capability is an important part of the system, since it verifies the algorithm and its implementation from the high-level design file input, rather than from the circuit layout or a lower level description, which would be less efficient.

The compiler is broken into two passes. The first pass extracts hardware parameters and assembles the microcode from the design file. The output of the first pass is an *intermediate file* which describes the macrocells at the level of the smaller library cells. A net list describes how the macrocells are interconnected. The intermediate file is human-readable to aid in program development.

The second pass of the compiler first assembles the macrocells, then interconnects them to form a mask-level description of the IC. In order to make the software technology independent, a special file (the descriptor file) contains dimensional information on the library cells and their terminals. A few additional parameters define the technology for the purposes of macrocell placement and signal routing. With the technology-specific information encapsulated in this way, the compiler may be used for any MOS process by redesigning only the cell library.

The cell library currently in use is for a 3-micron NMOS process, with a 3-micron CMOS cell library under development. The library contains a total of 160 cells. The cells are designed to operate at a 5 MHz maximum clock rate over all possible configurations of the macrocells.

## 2. Input-output and multiprocessor organization.

Signal processing IC's designed with the compiler will typically be used as peripheral chips to a host microprocessor. For this reason, two separate interfaces to the IC are provided (Fig.2).

Signal data is transferred each sample over a bidirectional signal data bus. Since the system clock rate is much higher than the signal sample rate, many such signals may be transferred each sample interval. These signals are the sampled-data inputs and outputs of the IC. Exactly one of the processors in the multiprocessor IC connects to this bus.

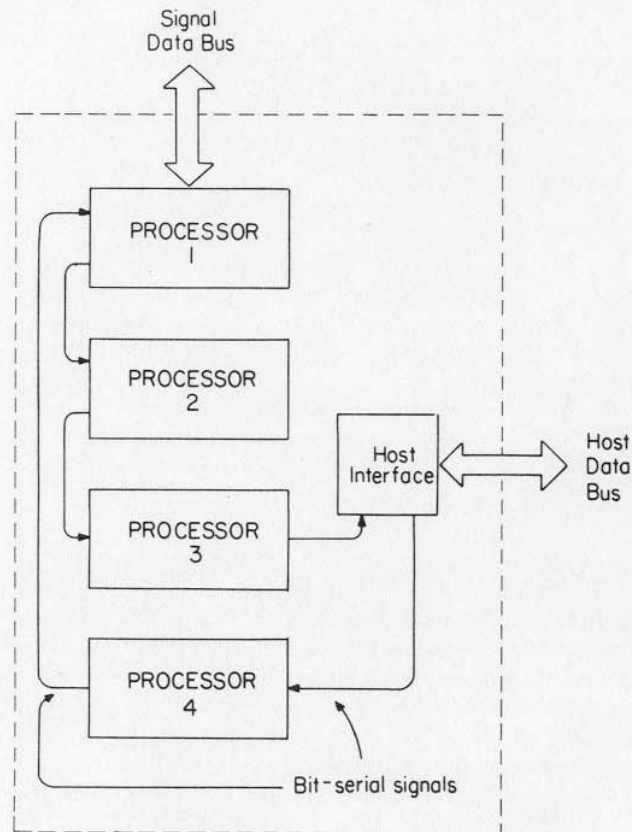


Fig.2 Organization of a four-processor IC

A separate host data bus connects the IC to its host microprocessor. Transfers over this bus are performed by the host in response to an interrupt from the signal processing IC. In general, a frame rate is defined, where a frame is an interval many samples in length. A block of data is transferred over the host data bus once each frame. This data is buffered in the IC in a host interface macrocell. Including the host interface hardware in the signal processing IC reduces the hardware costs of systems using the IC.

As shown in Fig.2, the IC contains some number of processors (from one up to perhaps four) and a host interface. The processors themselves contain several different macrocells as described in Section 3. An important aspect of the design of multiprocessor signal processor IC's is that the IC is dedicated to a single algorithm, and each processor is dedicated to a specific part of the algorithm. It is often natural to decompose a signal processing algorithm in this fashion. Because of the flexibility of the macrocell approach, the individual processors can be customized for their particular task.

Interprocessor communication is conducted over bit-serial data paths connecting the processors. The topology of this network is determined by the designer and is expressed in abstract form in the design file. The compiler then translates this topology into the requisite hardware, including parallel-serial and serial-parallel converters only where necessary. Thus it is advantageous to partition the algorithm such that interprocessor communication is minimized, since this reduces the amount of both circuitry and wiring.

### 3. Processor Architecture

The individual processors of the multiprocessor IC's are characterized by the following:

- (1) A control sequencer microprogrammed for a dedicated function
- (2) A bit-parallel, single accumulator arithmetic unit
- (3) A separate unit to perform address arithmetic
- (4) A data memory for local variables and constants
- (5) Use of a finite state machine for decision-making.

A study of algorithms within the target range indicated that the same general architecture, with suitable parameterization, can perform the algorithms efficiently. Thus the word length, the size of the data memory, the number of serial I/O ports, and the address arithmetic hardware are all configured by the compiler from declarations contained in the design file. The control sequencer is similarly parameterized, and contains a read-only memory programmed with microcode for the signal processing algorithm.

Each processor is organized into the following macrocells:

- PC (program counter)
- ROM (microcode read-only memory)
- SPC (subprogram counter - optional)
- AAU (address arithmetic unit - optional)
- FSM (finite state machine - optional)
- AUIO (arithmetic unit with I/O ports)
- RAM (processor data memory)

A fully configured processor is diagrammed in Fig.3.

The processor executes its microprogram (stored in ROM) once per sample interval. The microprogram consist of a *main program*, optionally followed by a fixed number of *subprogram* iterations. Except for this iteration pattern, there are no branches (conditional or unconditional) in the program execution.

The AAU is included only if address indexing is used. The address field of the control ROM is modified by the AAU to create the effective address for the data memory (RAM). Two index counters, IX and IY, may be included in the AAU. Indexing is used in signal processors to allow repeated sections of an algorithm to be multiplexed into a single piece of code.

The IX counter counts the subroutine iterations. IX equals -1 during the main program, 0 during the first subroutine iteration, 1 during the second iteration, and so forth. Indexing by the IX counter allows the subprogram to operate on elements of an array in data memory, accessing a different element each time the subprogram iterates.

IY is a sample counter with a fixed modulus. Indexing by IY allows the main program to operate on elements of an array in data memory, accessing different elements from sample to sample. This permits multiplexed processing of a group of decimated signals.

An additional option allows more general addressing, including pointer arithmetic. In each case, only the hardware needed to perform the types of addressing actually used are compiled into the AAU macrocell.

Viewed together, the ROM, PC, SPC and AAU present a stream of horizontal control and address words to the AUJO, RAM, and (optional) FSM macrocells. Signal data is processed in the arithmetic unit of the AUJO macrocell. RAM is used for data storage. FSM, if included, performs control and decision-making functions.

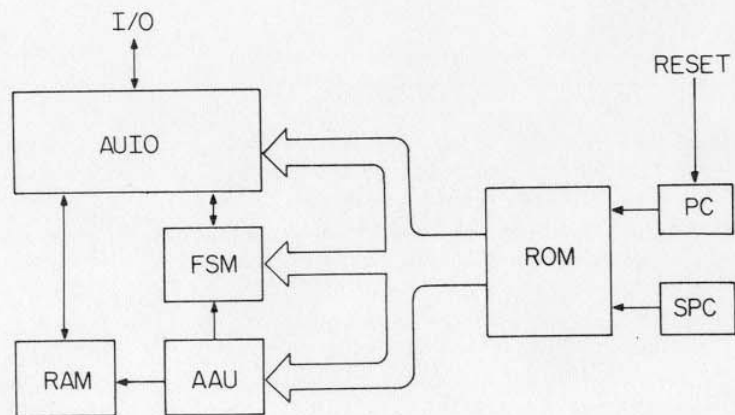


Fig.3 Macrocells forming a single processor

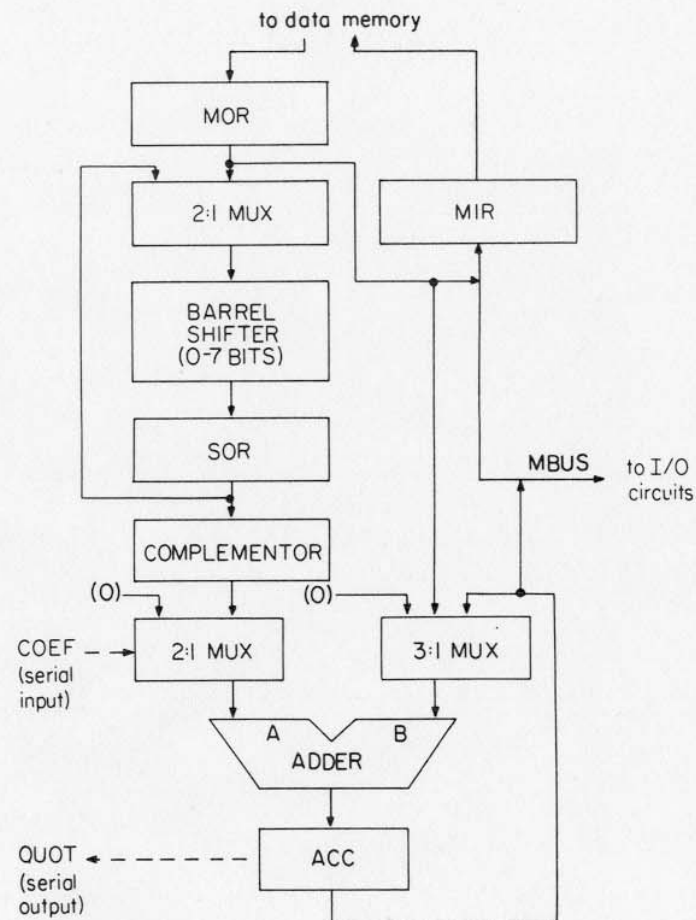


Fig.4 Processor data-path architecture

a saturating adder, three registers (MOR, SOR and ACC), and a transparent latch (MIR). This processor may be microprogrammed for all common signal processing functions.

Essential to any signal processor architecture is the ability to perform multiply-accumulate operations. Two cases are considered separately: multiplying signal data by a variable coefficient external to the processor, and multiplying signal data by a constant.

In the case of variable coefficients, the coefficient is available externally in a bit-serial, MSB first, fractional two's complement format. Each processor arithmetic unit allows two such serial coefficient inputs. To multiply a signal  $A$  by an  $n$ -bit coefficient  $k$ , the following equation is used:

$$kA = -A + \bar{k}_{n-1}A + k_{n-2}\frac{A}{2} + k_{n-3}\frac{A}{4} + \dots$$

Here  $k_i$  is the  $i$ -th bit of  $k$ . Each term on the right-hand side is a partial product, which is non-zero depending on one of the bits of the coefficient  $k$ . So to perform these variable-coefficient multiplies, the external bit-serial coefficient must be multiplexed into the control path for the arithmetic unit. This allows the partial products to be summed sequentially, one per clock cycle.

In the case of a fixed coefficient, it is not necessary to input the coefficient into the processor. Instead, a signed-digit representation of the coefficient is embedded in the control stream. This is done by specifying a sequence of shift depths and sign values. Thus, a fixed coefficient  $g$  has signed-digit representations of the form

$$g = (-1)^f 2^e 0 + (-1)^f 1 2^e + \dots$$

Any binary number has such a representation with the minimum number of digits. This is known as the canonical signed-digit (CSD) representation. The importance of the CSD representation is that it minimizes the number of clock cycles required to perform a multiply by a fixed coefficient. The multiply is performed by sequencing the barrel shifter and complementer so that it presents the desired terms to the accumulator.

In both the above cases, a sequence of multiplies may be performed, with the result of each being accumulated. This is an efficient feature of the single-accumulator architecture. The same accumulator is used for adding partial products, and for accumulating the results of several multiplies.

The arithmetic unit may also be microprogrammed to perform two-quadrant divide operations. Circuitry is included that makes the bit-serial quotient available in two's-complement form.

Generally the result of a multiply-accumulate operation is to be written back into the data memory. The transfer from ACC to the data memory is through the transparent latch MIR. By holding MIR transparent the transfer may be made immediately. Alternatively, MIR may store the data until the next free memory cycle. This use of MIR prevents congestion resulting from too frequent memory accesses.

The data memory itself may contain both read/write and read-only locations. The read/write locations are used to store state variables required for any signal processing. The read-only locations are used when it is necessary to introduce constants into the data path. Using the macrocell approach to designing the memory circuits supports this intermixing. The memory is configured

from the design file declarations of variables and constants.

The lack of conditional branches in the control flow does not allow a conventional approach to implementing decision-making operations. Instead, a PLA-based finite state machine (FSM) may be included as an adjunct to the data path. The programming of the PLA is specified in the design file, with the FSM's state variables corresponding to logical values in the algorithm. A conditional write operation is supported, which conditions the assignment of a value to a variable in data memory on the state of the FSM. This fairly simple decision-making capability satisfies the requirements of signal processors.

Each processor executes its microprogram once per sample interval. Since primitive operations such as multiplies must be microcoded, it follows that the sample rate must be substantially slower than the processor's clock rate. Otherwise, there would not be time for a significant amount of processing during the sample interval. On the other hand, if the ratio of clock rate to sample rate becomes very large, the architecture exhibits an imbalance wherein the control ROM consumes nearly all of the silicon area, and the data path only a small fraction. Based on the relative sizes of the library cells, the processor architecture is most efficient if the ratio of clock rate to sample rate is between 50 and 1000. This corresponds to a sample rate range of 5 kHz to 100 kHz, since the cell library is designed to operate at a maximum 5 MHz clock rate.

#### 4. Layout Generation

The first step in layout generation is the assembly of macrocells from the intermediate file description. This is done essentially by tiling of library cells. In the process, information on terminal locations is preserved for use as input to subsequent routing programs.

An important step in any macrocell-based design is the proper placement and interconnection of the macrocells. The silicon compiler does not require a general-purpose placement and routing program. Instead, it is only necessary to assemble those IC's the system is capable of producing. This more constrained problem is much more easily solved.

The problem is subdivided into two phases. In the first phase, the individual processors are assembled. In the second phase, the processors, the host interface (if included) and the bonding pads are assembled into the final IC.

A fixed-floorplan approach is used to assemble the processors from the separate macrocells. The floorplan consists of a left side (containing AUJO, RAM, AAU and FSM), a central wiring channel, and a right side (containing PC, ROM and SPC). The relative placement of the macrocells is defined by the floorplan, with absolute placement dependent on the sizes of the macrocells and the wiring channels. Wiring internal to either the left or right side is accomplished by repeatedly calling a simple river router.

The central channel is wired by a general purpose channel router, based on Kuh's channel routing algorithm [4]. The algorithm as originally stated in [4] required the terminals along either side of the channel to be aligned with a coarse grid, the dimensions of the grid being the minimum contact-to-contact spacing. The algorithm was modified to allow terminals to be of arbitrary position along the edges of the channel.

The single, fixed-floorplan approach does not always produce an efficient placement. Additional floorplans are being developed, and the software has the ability to select the most compact floorplan from a small number of alternatives.

Once the individual processors are assembled, the IC itself is assembled from the processors, the

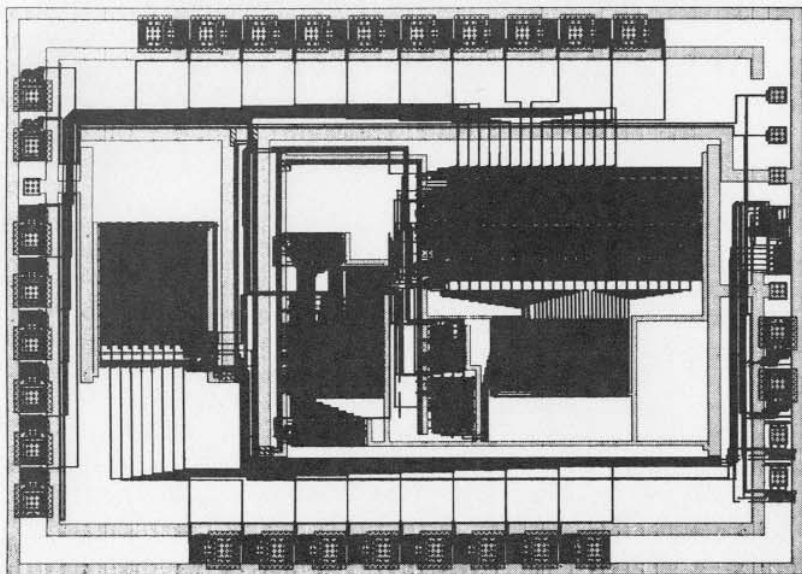


Fig.5 Circuit plot of the audio equalizer (3.6 x 5.1 mm)

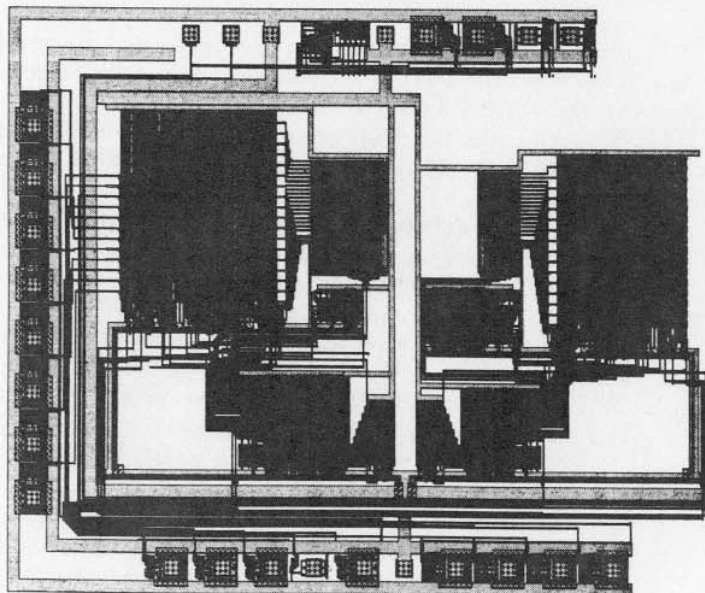


Fig.6 Circuit plot of the decision feedback equalizer (3.7 x 4.5 mm)

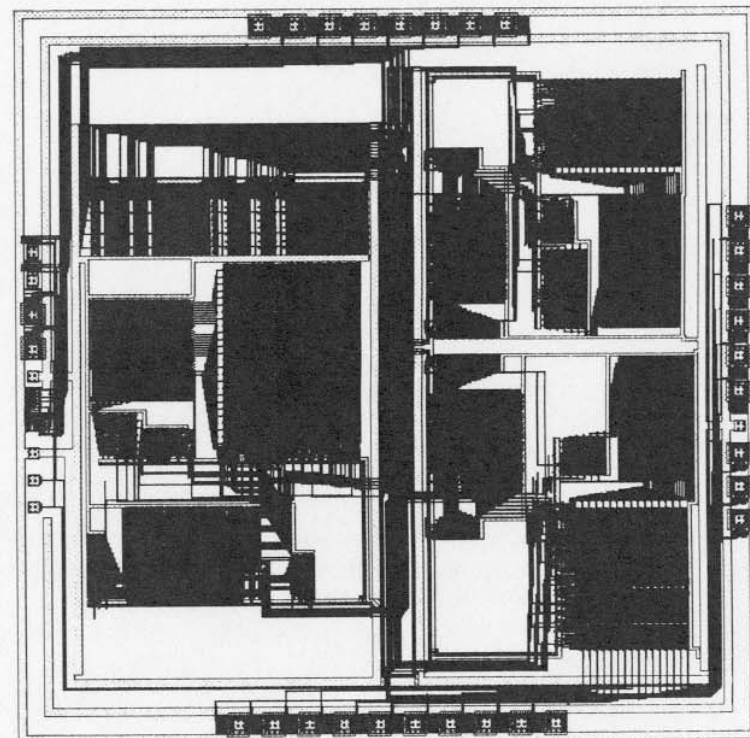


Fig.7 Circuit plot of the LPC vocoder (7.0 x 7.0 mm)

host interface and the bonding pad arrays. The current approach to this global placement and routing is to partition the blocks into two columns, determining an optimal partitioning, ordering and rotational orientation for the various blocks. The channel routing routine is then called up to five times to complete the wiring of the IC.

More sophisticated global placement and routing routines are being studied.

## 5. Examples

Design file inputs have been prepared for a number of applications. Figs. 5-7 shows circuit layout plots of three of these, with the following characteristics:

function	processors	transistors	sample rate
audio equalizer	1	6,900	50 KHz
LPC vocoder	3	27,400	8 KHz
decision feedback equalizer	2	8,300	116 KHz

The macrocell-based silicon compiler is still actively being researched. However, it is possible at this point to draw a few general conclusions.

The macrocell approach can generate complex signal-processing circuits with far less effort than manual circuit design and layout. A few successful applications could easily amortize the development costs for the compiler and cell library. Finally, and most importantly, the design files can be prepared by signal processing researchers who have no prior knowledge of integrated circuit design. This one fact significantly expands the range of individuals who may now involve themselves in custom digital signal processing IC design, a field previously reserved for a select few.

## Acknowledgements

The authors would like to thank Peter Ruetz, Jeremy Tzeng and Mats Torkelson for their contributions to this project. Research supported in part by the Defense Advance Research Projects Agency, Contract No. MDA903-79-C-0429.

## References

1. S.S.Magar, E.R.Caudel, A.W.Leigh, "A Microcomputer with Digital Signal Processing Capability", *International Solid State Circuits Conference Digest*, San Francisco, 1982, pp. 32-33.
2. S. Pope, *Macrocell Design for Signal Processing*, Ph.D. Dissertation, University of California, Berkeley, CA, December 1984.
3. P.B.Denyer, D.Renshaw, "Case Studies in VLSI Signal Processing Using a Silicon Compiler", *Proc., International Conference on Acoustics, Speech and Signal Processing*, 1983.
4. T.Yoshimura, E.S.Kuh, "Efficient Algorithms for Channel Routing", *IEEE Transactions on Computer Aided Design*, V. CAD-1, Jan. 1982, pp. 25-35.

**Stephen P. Pope** received the Bachelor of Science degree from the California Institute of Technology, Pasadena, California, in 1978, and completed his doctoral work at the University of California, Berkeley, California in 1984. He is currently on the engineering staff of Cyclotomics Corp., Berkeley, California, where he is studying IC implementations of error detection and correction algorithms.

**Jan Rabaey** was born in Veurne, Belgium on August 15, 1955. He received the E.E. and the Ph.D. degrees in applied sciences in 1978 and 1983, respectively, from the Katholieke Universiteit Leuven, Belgium, where he worked on computer aided design tools for switched capacitor circuits. Since 1983 he has been a visiting research engineer at the University of California, Berkeley. His current interests are in computer aided analysis and automated design of digital signal processing circuits.

**Robert W. Brodersen** received the B.S. degree from California State Polytechnic College, Pomona, in 1966, and the E.E., M.S., and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1968, 1968, and 1972 respectively. From 1972 to 1976 he was with Texas Instruments, Inc., Dallas, studying the operation and application of charge-coupled devices. In 1976 he joined the faculty of the University of California, Berkeley, where he is a Professor. He is studying the application of IC technology to signal processing, image processing, and user interfaces.