Signal processor chip implementation

by Jean Paul Beraud

Very large microprocessors can now be integrated on a single chip; the integration eliminates packaging delays and is especially attractive for performance-oriented processors such as signal processors. This paper describes a semicustom signal processor chip designed jointly by the IBM France Essonnes and La Gaude laboratories. The logic is implemented from an optimized library of bipolar circuits. Layouts are compatible and designed to map data flow structures efficiently. Chip design time has been greatly reduced through the use of developed CAD tools tailored to our methodology. The design achieves twice the density that would be possible (with the same technology) with a masterslice. The chip's high performance has been verified with hardware; it provides enough computation power for 125 second-order filters with 8-kHz sampling of the input signal.

Introduction

Very complex chips such as signal processors are now possible due to the evolution of the technology and improved tools to facilitate its development. There are two basic design methodologies, gate array (also called masterslice) and custom design. In gate array design flexibility is very high; the physical design proceeds rapidly, requiring few manual interventions. Manufacturing time is also shorter because processing comprises only personalization steps (metallurgy). To facilitate these aspects, a compromise was made in gate array design and the result is that the density is not optimized. Custom design is the opposite of the gate array design; it optimizes the silicon density but requires a considerable amount of manual effort.

The IBM Signal Processor (SP) chip described here uses a mix of the two methodologies; it is a high-density chip which requires little manual effort to produce. The data-flow-oriented structure of the signal processor permits use of a macro element structure to minimize the wiring. Performances of more than 10 MIPS (millions of instructions per second) are reached on a chip 25 mm square, with a density for some internal parts up to 250 gates per square millimeter.

This paper gives a brief review of the architecture, details on building blocks used, some choices made concerning logical design implementation, the physical design methodology, and hardware results.

Architecture

The SP is characterized by separate instruction and data memories, one bidirectional data bus, a single instruction format which contains two independent operation codes, and the use of a fast parallel multiplier [1].

The SP (see **Figure 1**) comprises three main parts which operate simultaneously:

1. *The sequencer* controls the instruction memory. It performs a sequential increment of the instruction address or forces an instruction address at a value given by the program during branches. The computed address is stored in the Instruction Address Register (IAR) at the end of the cycle. Interrupt management is also controlled by the sequencer.

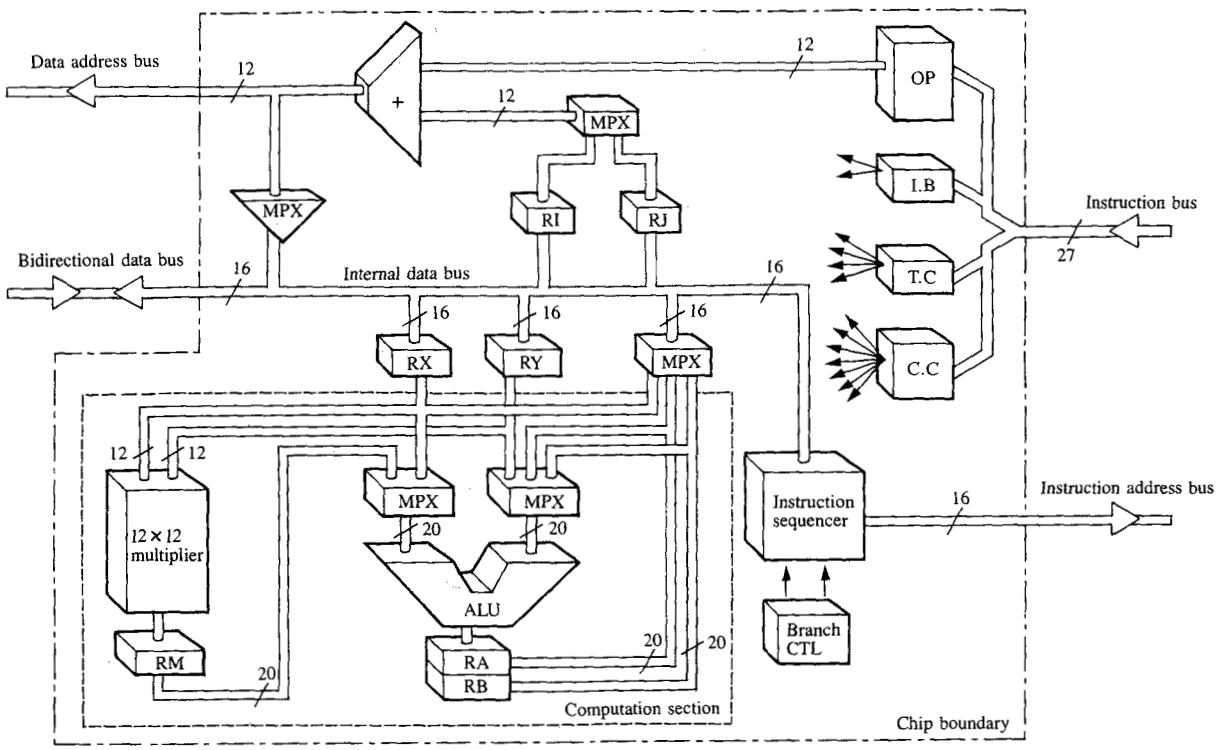

Figure 1

Signal processor block diagram.

2. *The address generator* computes the address of the data memory by utilizing one of the two index registers (RI, RJ). This computation is made in the same cycle as the instruction fetch, and the result is stored in the Data Address Register (DAR) at the end of the cycle.
3. *The computer portion* is composed of the ALU and the multiplier. Both have two identical input registers, RX and RY. Multiply and ALU operations are executed in parallel. The multiplication is pipelined in two cycles and the result is stored in the multiplier output register (RM). The ALU is accumulator-based, using the two accumulators RA and RB. The ALU operates with two of the five possible stored values, RX, RY, RM, RA, and RB. Correct synchronism is required between ALU and multiplier operations; nevertheless the logic design provides for this multiplication and accumulation in one cycle.

Circuit library

The chip comprises a library of predesigned building blocks. A Transistor-Transistor-Logic (TTL)-compatible circuit family was chosen for the following reasons:

- Ease of interface with IBM bipolar logic masterslices (power supplies and logic level).
- A good ratio for power/performance and area trade-off for the range of performance.
- Ease of use in logic design.

The basic gate (**Figure 2**) is a Schottky antisaturated TTL gate operating from a single 1.7-V power supply; this allows a power consumption of 0.32 mW for a typical delay of only 1.9 ns. The standard 2.5-μm IBM technology which was used features oxide isolation, a 2-kΩ ion-implanted resistor, and three levels of metallurgy.

Table 1 lists the basic logic blocks with their main characteristics. All of the circuits are compatible, and extensive computer simulations were performed to derive a set of wiring rules defining all of the allowed configurations. Delay equations were also established by simulation. All of the rules were coded for use in computer-aided design programs performing logic simulation, net validity checking, and delay calculation.

A separate custom circuit design and layout were created for every basic block. The savings made in component

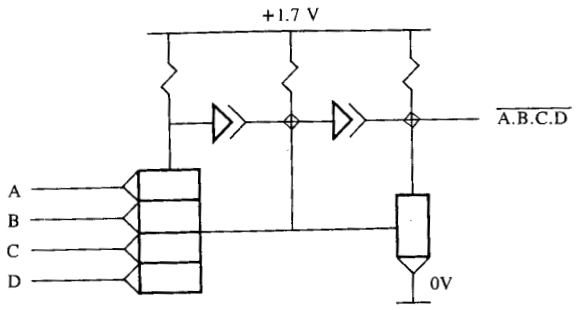

Figure 2

Basic TTL gate.

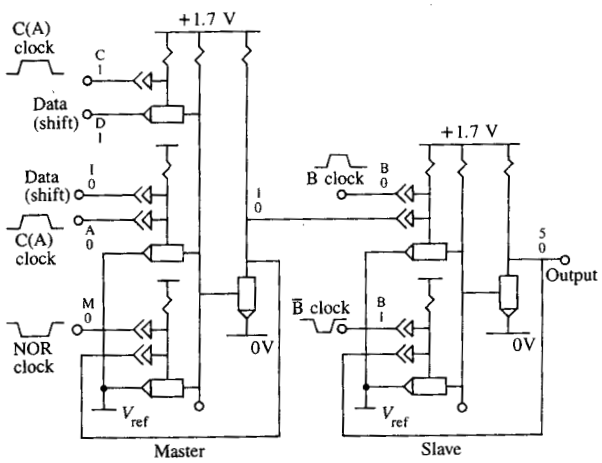

Figure 3

Two-input master-slave latch.

Table 1 Characteristics of the main building blocks.

Building blocks (type)	*Power* (mW)	*Delay* (ns)	*Area* (μm^2)
4-way AND-INVERT	0.32	1.87	105·35
8-way AND-INVERT	0.32	2.32	210·35
2-way AND-INVERT	0.32	1.62	105·35
AND-OR-INVERT	0.55	1.62	105·49
EXCLUSIVE-OR	0.55	2.06	105·63
Fast 4-way AND-INVERT	0.64	1.21	105·35
9-bit parity	7.4	5.38	9·105·77
1-input latch	0.66	2.28	105·56
B clock register driver	4.3	Positive output: 1.7	210·56
C clock register driver	2.3	2.45	105·42
Off-chip driver	4.3	At 30 pF:10	105·189
Off-chip receiver	1.15	1.34	105·77

count, compared to the same implementation in TTL NAND gates, can be seen in the schematic of **Figure 3**, which represents a two-input master-slave latch.

Physical library

To ensure topological compatibility among the various blocks, they were laid out with the following constraints:

- Use of a common horizontal pitch.
- Power distribution through two vertical buses (1.7 V, Gnd) at fixed locations on each side of the block.
- Circuit layout using only first-level metal, leaving 11 second-level wiring channels per pitch free and clear.
- I/Os connected to vias placed on second-level wiring channels. Multiple I/O locations are offered for every logic book I/O.

The physical structure is designed to map data flow logic structures efficiently into data flow macros. In those macros the common horizontal pitch of the block represents a bit pitch; the control wiring is directly embedded in the block at first level in such a way that its ordinate is the same at each vertical edge of the macro (**Figure 4**). When this is done, the hierarchical structure of macros becomes obvious. For example, one can easily create a register with its control clock by simple placement of latches side by side on the same horizontal line. Complex data flow macros can be built by stacking such rows of building blocks. The power supply distribution also matches the bit pitch. In order to save silicon area, adjacent blocks are mirrored (rotated) about the $+V$ distribution bus. If the adjacent blocks are not identical in function, the design is adjusted to allow sharing of the $+V$ resistor contacts.

Figure 5 represents the layout of the two-input master-slave LSSD latch. Only metallurgy and via levels are shown on the figure.

Logical design

By using blocks defined in the technology library, the logic design is made to conform as closely as possible to the architecture. It must also respect speed requirements and constraints related to the packaging (e.g., pins, module size, simultaneous switching). Throughout this text, logic complexity is measured in three-input NAND equivalent gates and represents an equivalence of 5000 gates.

The following sections provide more detail on specific logic design topics.

- *Multiplier algorithm*

The multiplier is a large portion of the chip in terms of circuit count (2000 gates). It is also a critical part because of the long data path associated with the cascade adders. To minimize circuit count and data path length, the modified Booth algorithm was proposed. It uses fewer circuits than the

conventional algorithm measured bit by bit, though it is larger in terms of silicon area. The reason for this is as follows: In a matrix generated by adders, the lines of adders are not the same length. The first line has 13 adders, and the number increases to 18 adders on the sixth line. This configuration is not well adapted to the stack design, where the same length for each line is preferable. This is the case in the conventional multiplier, where 12 adders are used in each line.

For the modified Booth algorithm, the data path is shorter, but like the conventional algorithm it requires a pipeline register to meet the speed requirement. It also requires more random logic per stage for decoding actions. Unfortunately, the macro stack design used is very poor in terms of density for random logic.

Thus, the unequal number of adders per line, no possible gain with the pipeline register, and the low density of the random logic combined to force the elimination of the Booth algorithm for the multiplier. The conventional bit-by-bit multiplier algorithm was implemented instead.

• *Multiplier adder cell*

In the multiplier, the basic cell is the adder. It was designed as shown in **Figure 6**. The odd adder lines provide "carry-out (n)" out-of-phase and use "carry-in ($n-1$)" in-phase, while the even adder lines provide "carry-out (n)" in-phase and use "carry-in ($n-1$)" out-of-phase. This mix allows the Carry branch and the Sum branch to be equivalent in block delay, and it also gives an average circuit count of 5 per adder.

• *Simultaneous switching*

The SP has many types of buses: data, instruction word, RAM address, and instruction address. The last two are output buses of 12 and 16 bits, respectively. All data are taken from registers before going through the output drivers. Because these registers are clocked with the same master clock, the 28 drivers could switch simultaneously and generate noise which might have an impact on the internal circuits. For this reason, the clock of the RAM address bus register is delayed 5 ns relative to the master clock. Twelve of the 28 drivers switch 5 ns later; they no longer switch simultaneously with the 16 other drivers, and switching noise is thereby avoided.

• *Miscellaneous*

Some other implementation options were chosen for reasons of speed:

- Carry-look-ahead adder for RAM address generation (Index plus Operand);
- Duplication of the input registers X and Y, one set on the ALU input and the other set on the multiplier input;

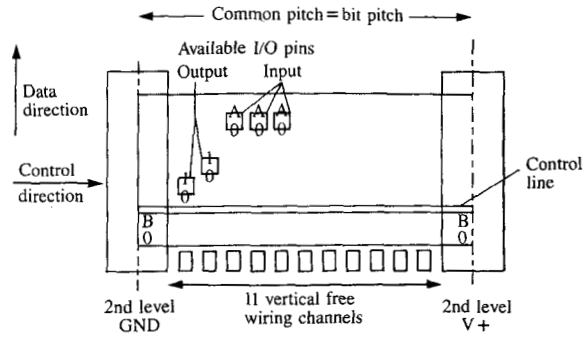

Figure 4

Building block structure.

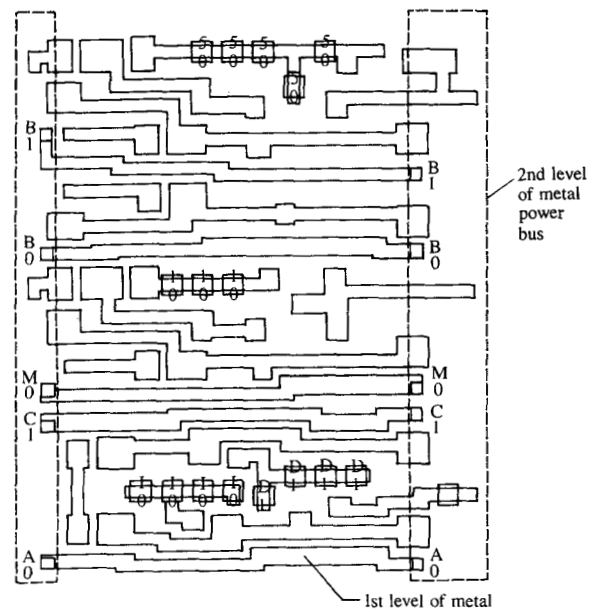

Figure 5

Two-input master-slave latch.

- Double internal data bus, one part for the ingoing data, the other for the outgoing data.

• *Simulation*

The logic description coded in a computer was treated as a real processor. Patterns representing instruction word and input data were applied to the pins of the chip in synchronism with functional clocks. A complete instruction was simulated during each step clock. Several routines were

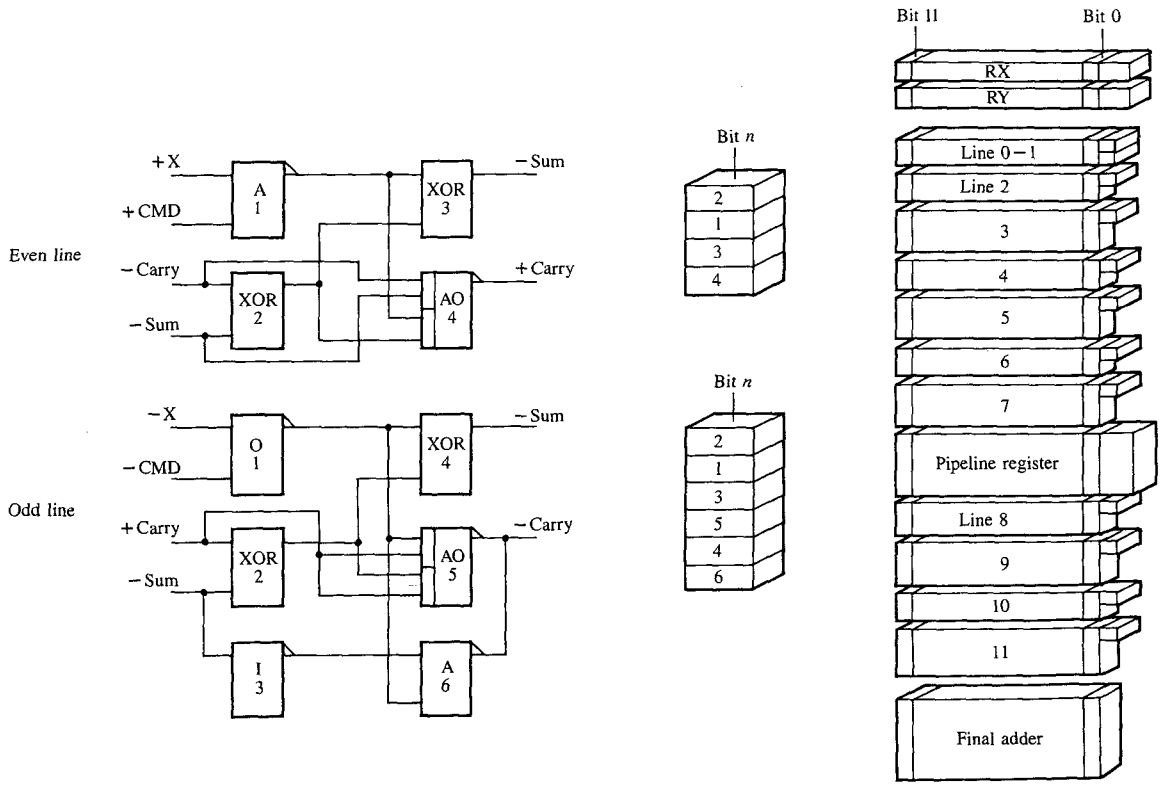

Figure 6

Multiplier basic cell and adder line organization.

executed in this manner to exercise different parts of the logic with as much data as possible. All input configurations were used to simulate various possible external events, mainly ones related to interrupts. A total of about 500 patterns (one pattern = one complete set of inputs during one clock cycle) were used to validate the chip design.

• *Design for testability*

In addition to reflecting the architecture, the logic implementation must provide for the final test product requirements. The product is considered to be completely tested if all internal gates have been switched through two states and if the event is detectable at an output I/O pin. To test the chip easily without too many extra test pins, the LSSD (Level-Sensitive Scan Design) concept (**Figure 7**) was used. It requires that all memory elements (latches) be wired together serially to form a shift register. Each latch can thus be used either as an input or as an output for the testing of combinatorial logic. For a master-slave latch design it requires at each master latch an extra input controlled with a shift clock in order to load and unload the created shift register (Fig. 7). A minimum of four extra pads is required: the two shift clocks, the scan data input, and the scan data output. Thus, the total LSSD overhead represents 4% additional gates, 5% additional power, and 3% additional pads.

Physical design methodology

The physical design methodology is described in another paper [2]. Its key feature is the wiring of data flow macros. In such macros, block placement is done manually in terms of rows and columns, and wiring is performed through a locally developed program [2]. The inputs of the program comprise the coded logic description, the digitized layouts of the blocks, and the relative placement of the blocks (row, column).

The wiring algorithm prioritizes the vertical direction (second level of metallization) for data wiring. The horizontal direction is mainly used for controls, which are already embedded in the blocks whenever possible. The program determines the number of horizontal channels needed between rows to wire the macro without overflows so that inter-row spacing is dynamically adjusted to demand (**Figure 8**).

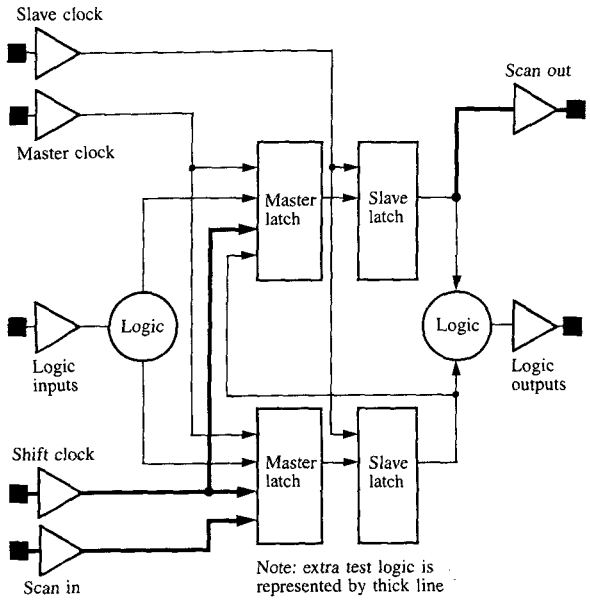

Figure 7

LSSD latch concept.

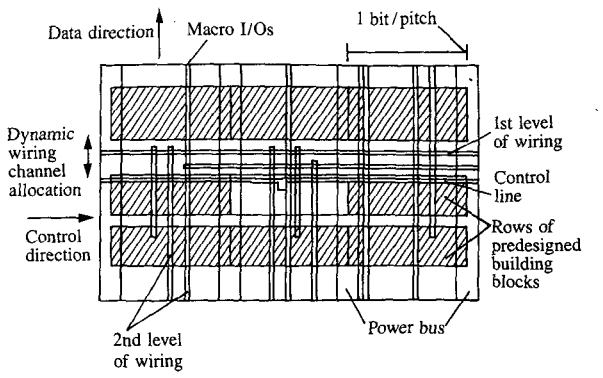

Figure 8

Dataflow structure.

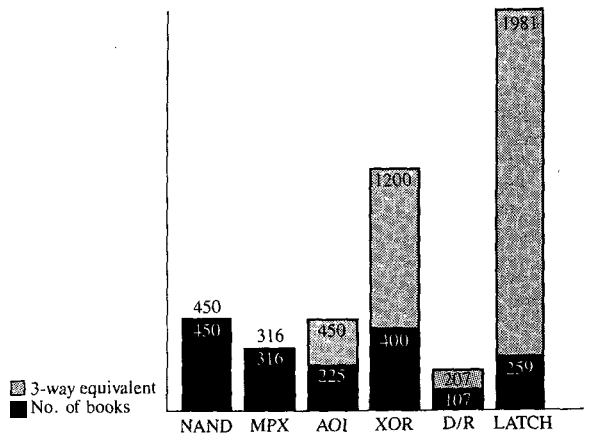

Figure 9

Logic gate distribution.

The outputs of the wiring program include physical data allowing direct generation of masks for production and electrical data for net capacitance and voltage drop.

Even though this program is tailored for data flow macros and is less efficient for random logic macros, it was used for control wiring. This did not create a wiring penalty and allowed the data set structure for the data flow and control portions to be the same. Only 150 gates out of 5000 were required for controls. With this methodology it took only 0.66 man-year to perform the complete chip design. No errors were found on the first hardware realization.

Main characteristics of the chip

The chip contains 1932 blocks equivalent to 5000 three-way NAND gates mapped on a 4.3 × 6.3-millimeter chip. **Figure 9** is a histogram of the usage of the various elements of the library with their equivalent gate count in three-way AND-Inverters. All latches are LSSD master-slave latches for testing purposes. This explains the high equivalent gate count for latches.

All data flow macros have densities higher than 200 gates per square millimeter. The chip was wired into six data flow macros (**Figure 10**): multiplier (three macros), ALU, sequencer, and address generator. The random logic area represents only 6.6% of the chip area; another 10.7% is used for global wiring or remains unused.

The third level of wiring is used for power distribution and connection to the I/O pads. The pad footprint is a 17 × 17 matrix with the center pads used for power distribution and 88 external pads for I/O. The chip is mounted on a 28-mm metallized ceramic module having 1.6 W power dissipation requiring a forced air cooling environment.

Figure 11 is a microphotograph of the chip environment.

Hardware results

The chip has two critical paths; one is purely internal and appears on the first half of the multiplier (before the pipeline register). In the worst-case data configuration, the typical minimum cycle time is 60 ns for the multiplication. The second path depends on the instruction memory access time. The 60-ns typical time can be maintained if the typical access time of the instruction memory is 18 ns. If the

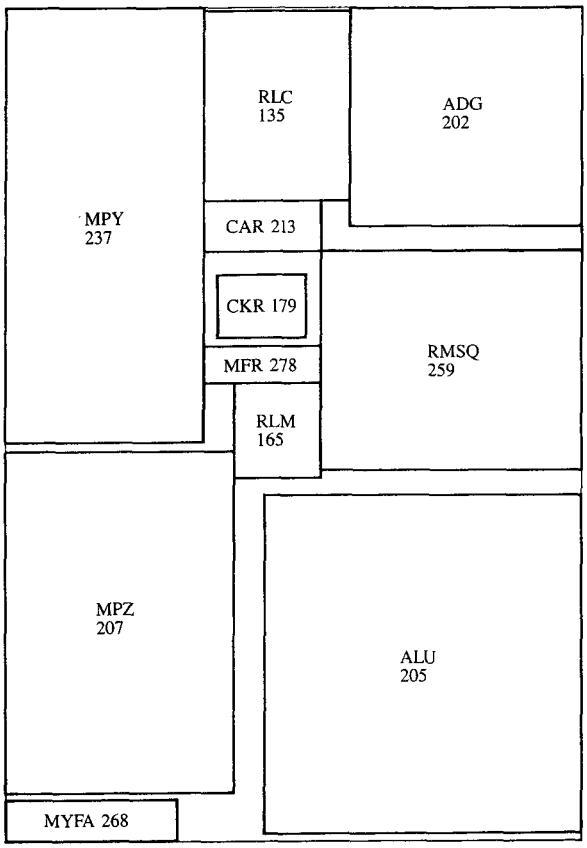

Figure 10

Average density (in three ways AI-equivalent).

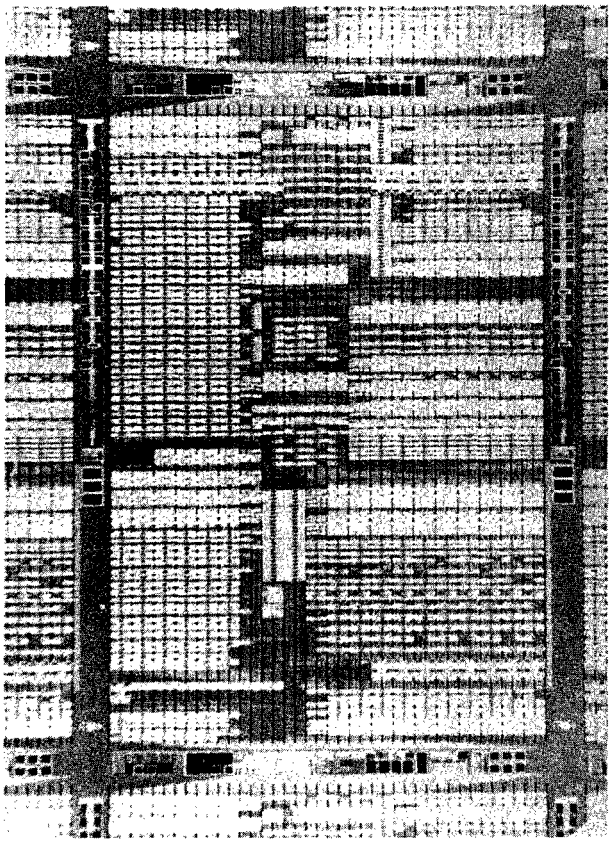

Figure 11

Signal processor chip.

number is 30 ns, the chip cycle time must be increased accordingly to 72 ns.

Conclusions

With a library of dense circuits and good design tools, we have realized a very efficient circuit. Significant gain in silicon area was obtained by using a dataflow macro concept, 200 gates per square millimeter versus 100 gates per square millimeter for the conventional gate array. A gain in speed was also achieved by minimization of wiring capacitance and optimization of the multiplier implementation by using the double-cell adder. Its high speed and good density make the IBM SP a competitive processor chip for performing real-time signal processing and satisfying new needs in specific domains such as radar, sonar, robotics, guidance, control, communications, audio, and imaging.

References

1. G. Ungerboeck, D. Maiwald, H.-P. Kaeser, P. R. Chevillat, and J. P. Beraud, "Architecture of a Digital Signal Processor," *IBM J. Res. Develop.* **29**, No. 2, 132–139 (1984, this issue).
2. M. Bauge, M. Richarme, and B. Vergnieres, "A Highly Automated Semi-Custom Approach for VLSI," *IEEE J. Solid-State Circuits* **SC-17**, No. 3, 465–472 (June 1981).

Received March 27, 1984; revised October 11, 1984

Jean Paul Beraud *IBM France, B.P. 58, 91102 Corbeil-Essonnes, Cedex, France.* Mr. Beraud is currently a logical designer engineer at the Essonnes laboratory. He joined IBM in La Gaude, France, in 1966 and worked with modem and audio numeric groups in the Advanced Technology Department. Since 1979 he has worked in the field of signal processing. In 1982, he joined the Component Development Laboratory in Essonnes to work on chip design. He received his Engineer Diploma from the Conservatoire National des Arts et Métiers de Nice, France, in 1971. Mr. Beraud received an IBM Outstanding Innovation Award in 1982 for the development of a real-time signal processor chip.