Anthony Correale

Physical Design of a Custom 16-Bit Microprocessor

The physical chip design aspects of a 16-bit, single-chip, custom-macro-designed microprocessor are described. This microprocessor represents the IBM System Products Division's highest-density VLSI FET processor design to date. The chip is a complex arrangement of over 6500 VLSI circuits utilizing a state-of-the-art polysilicon-gate HMOS-I (high-performance MOS) technology. The physical design of this chip required the use of a comprehensive methodology, from conception through completion. The methodology used in the design of the microprocessor was based on a hierarchical approach and is presented in this paper.

Introduction

Microprocessors are generally used in IBM products to minimize development cost by providing a common building block that can be programmed to meet a wide range of needs. The high development cost of microprocessors makes such a strategy important, since the development of a new processor for each new product would result in high product costs.

In the early 1970s, a strategy to build a family of upward-compatible microcomputers, known as "universal controllers" or UCs, to be used in general-purpose and industry-specific products, was conceived and implemented [1]. At that time there were few alternatives available from outside sources. Since then, many semiconductor manufacturers have produced their own microprocessor designs, including IBM.

This paper describes the physical design methodology which was used to produce a new custom-macro-designed microprocessor at the lowest development cost possible while meeting schedule, performance, and quality objectives. In the following sections, an overview of the microprocessor design is first presented, including its basic characteristics, the logic control and data flow, the macro design concept, and the physical layout of the chip. Then the method of logic partitioning into macros, technology selection considerations, the hierarchical approach used, and the use of a circuit design library are discussed. This is followed by descriptions of the dc circuit design methodology, the physical layout design considerations, the interactive graphics tools used in the macro and chip designs, and the verification techniques used for dimensional and logical-to-physical checking. Finally, the results of the design effort and some significant conclusions are presented.

Microprocessor design overview

The pertinent characteristics of the custom 16-bit microprocessor are listed in Table 1. Figure 1 shows the logic flow, where the various functions indicated have been implemented as macros. These can be considered as either Programmable Logic Array (PLA) macros (partitioning circuits, AND array, OR array, and output buffer circuits) [4] or random macros (registers, multiplexers, parity trees, etc.) [5]. The complexity of the macros ranges from 50 to 500 equivalent logic gates. (An equivalent logic gate is defined as a 2.5-way NOR; that is, 3.5 devices comprise an equivalent logic gate.)

Figure 2 illustrates the physical layout of the chip, showing the macro boundaries. The macros identified with an asterisk are those which make up the decode and control section shown at the left in Fig. 1. Of the 22 PLA macros which make up approximately one-half the circuits on the chip, the decode and control section was physically implemented using 18 of the total 22 PLA macros and 6 random logic macros. As seen from these figures, the decode and control section is a complex arrangement of many macros, whereas the other functions are implemented by single macros. Examples of functions implemented by single macros are the Cache and the Interrupt. These are shown in Fig. 2 as macros R40 and I30, respectively.

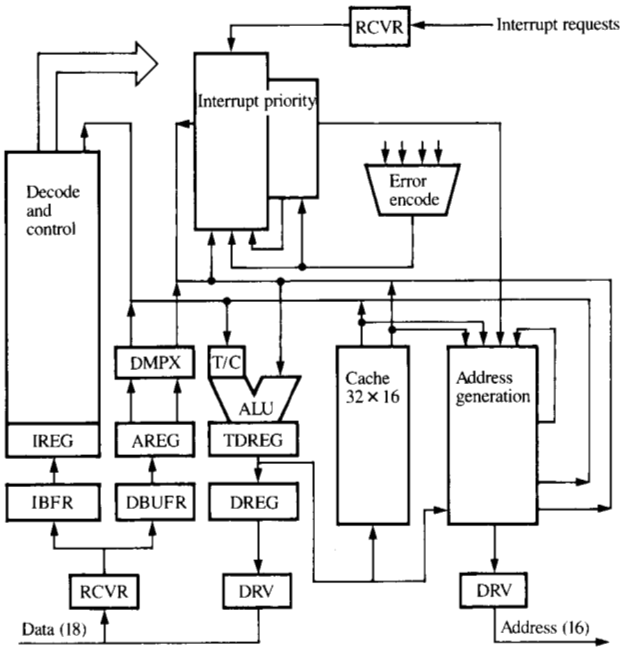

Figure 1 Logic control and data flow. (Note: AREG = A Register, DBUFR = Data Buffer, TDREG = Temporary Data Register, DREG = Data Register, IREG = Instruction Register, IBFR = Instruction Buffer, RCVR = Receiver, DRV = Driver.)

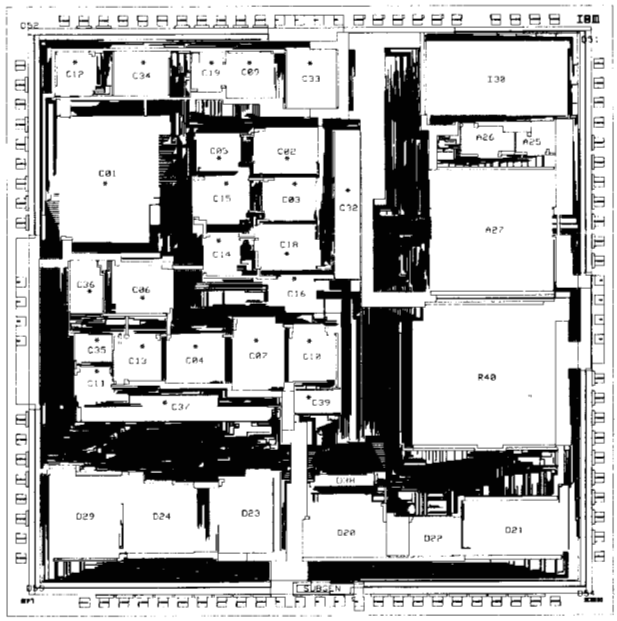

Figure 2 Microprocessor physical layout. (Areas identified by the alphanumeric designations are macro areas. Those marked with an * are those which form the decode and control function of the processor. Other functions are implemented by single macros.)

Logic partitioning into macros

Logic partitioning was performed initially on a functional basis. The logic gates were grouped into macros which implemented various logical functions such as address generation, the arithmetic and logical unit (ALU), interrupt priority, and error checking. The physical size of each macro was then determined. Images representing the size and shape of each macro were then located on a chip image. The macro placement was a function of the intermacro communication, the required overall chip performance, and the size and shape of the remaining available chip area.

In some instances, logic had to be repartitioned from a single macro into several macros having the size and shape of the chip area available. The logic partitioning into macros was therefore somewhat of an iterative process.

Technology considerations

The complexity of this product required a state-of-the-art technology for its implementation. A high-performance polysilicon-gate HMOS-1 FET technology with 3.5-μm minimum artwork geometries proved sufficient to meet all design objectives. These included chip size, performance, power dissipation, cost, and reliability.

The use of the polysilicon-gate technology enabled macro wiring to be placed directly over (on top of) the circuits,

Table 1 Custom 16-bit microprocessor characteristics.

Technology	HMOS-1
Design	Custom macro
Chip size (periodicity)	7.1 mm
Module	28 mm (multilayer ceramic, wire bond) [2]
Signal I/O pins	79
Machine cycle	500 ns, 1.70 μs per instruction (average)
Cooling	Convection, air
Power	800 mW (worst case)
Storage addressability	64K bytes, extendable to 512K bytes
I/O	Memory-mapped I/O, register-mapped I/O Eight interrupt levels under program and I/O control
Internal data path	16 bits
ALU width	Byte
On-chip RAM	32 bytes
Number of instructions	91
Total macros	42
Random macros	21*
PLA macros	22
Mask levels	10
Wiring levels	Metal Poly Diffusion
Devices	22781
Circuit types	82
Equivalent circuits	6509 (assumes 3.5 devices per circuit)
Percent testability [3]	99.8%

*One PLA macro is contained in a random macro.

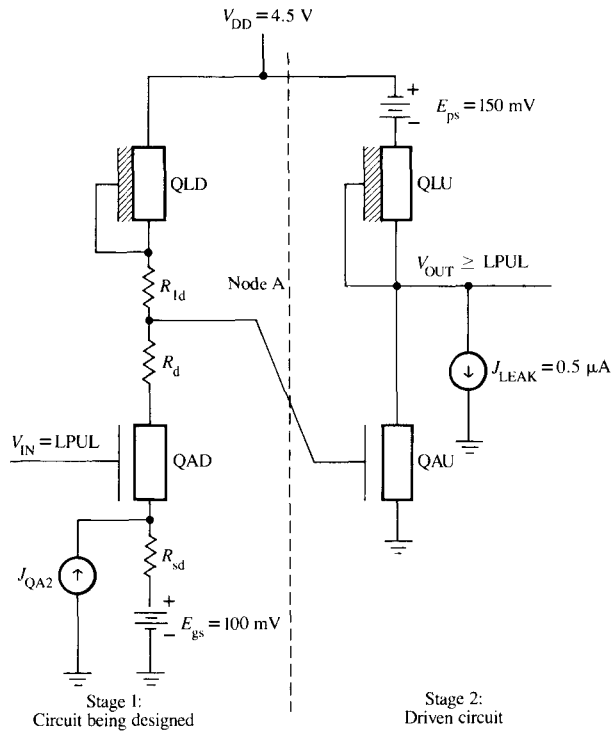

Figure 3 Two-stage open-loop circuit design

thereby saving significant chip area. This was possible because the circuit implementation required little or no metal. In a design as complex as that of the 16-bit microprocessor, every available piece of silicon real estate must be judiciously utilized. The ability to wire over the top of the circuits rather than placing the wire alongside the circuits, as is customarily done, saved precious area and made the implementation of the microprocessor possible on the required chip size.

Hierarchical approach

A hierarchical approach to the physical design of the custom 16-bit microprocessor was adopted whereby each subassembly was determined to be physically and logically correct prior to its integration into the next-higher-level subassembly by a variety of software checking programs which are described. This approach reduced the final chip checking to a more manageable task and allowed the achievement of a successful two-pass design.

A subassembly in this context can be an individual transistor, circuit, or macro—a macro being defined as an interconnection of circuits to perform a specified logical function such as instruction decode or address generation. Examples of macros are PLAs [6], registers, and configurations of random logic. The highest level of integration is the completed chip, which can be regarded as the ultimate macro.

Circuit library and circuit design

Once technology selection had been completed, the next step was to define and design a finite collection of circuits for the implementation of the microprocessor. Discussions with the logical designers led to the definition of the circuits on the basis of logical function, performance, power dissipation, and physical size. The circuits were defined in two categories, PLA and random logic. The PLA circuits were those used in the creation of PLA macros, namely, input partitioning, AND array, OR array, and output buffers. The random logic circuits were defined as all other circuits. Examples of random logic circuits are latches, receivers, off-chip drivers, unit-logic circuits, and various buffers.

Since the intent was to create a library of circuits which could be used for this project as well as for future or follow-on designs, a well-defined circuit-design methodology was established to ensure that all circuits were designed consistently under the same criteria (to be described in subsequent sections). This ensured that each circuit designer developed circuits compatible with those of other designers and allowed for the incorporation of new circuits for future products.

DC designs

The dc circuit design was performed using a two-stage open-loop technique, illustrated in Fig. 3. The circuit being designed, Stage 1, is comprised of the following: QLD and QAD, which are the load and active devices, respectively; R_{ld}, R_d, and R_{sd}, which are the parasitic resistances associated with these devices as well as any additional resistance allowed for circuit logic function expansion by dotting devices; J_{QA2}, which is a current source to simulate shared-source diffusions, as is the case in many PLA circuits; and E_{gs}, which is a 100-mV dc voltage source to simulate ground shift and noise.

The output, labeled as Node A, is fed directly into the input of the second stage, which is the driven circuit. The driven circuit is made up of the following: QLU and QAU, which are the load and active devices, respectively; E_{ps}, which is a 150-mV dc voltage source to simulate power supply distribution loss and noise; and J_{LEAK}, which is a current source to simulate 0.5 μA of leakage current for up-level degradation.

The input voltage level of the first stage is the Least Positive Up Level (LPUL) for the internal chip circuits. The output voltage at the intermediate node (Node A) must be sufficiently low (typically <0.6 volts) to ensure that the output of the second stage will be greater than or equal to the LPUL. This requirement ensures propagation of valid logic levels. The required voltage level at Node A determines the active device (QAD) width-to-length ratio of the first stage, and hence the dc design.

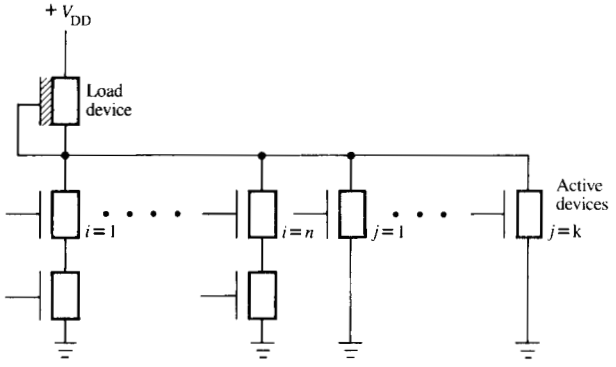

Figure 4 Pertaining to the definition of circuit "gain."

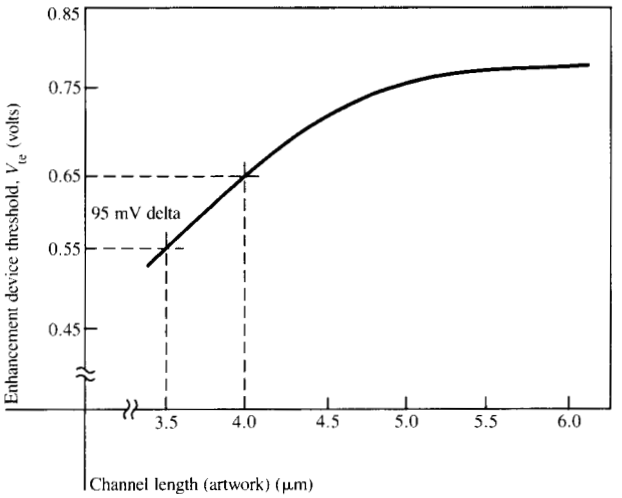

Figure 5 Enhancement device threshold *versus* channel length.

Since there was significant variation in different circuit types (width/length ratio, power dissipation, and logic function) a parameter called "gain" was defined to specify the compatibility of all circuits in the design.

With reference to Fig. 4, the *gain* of a circuit is defined as

$$Gain = \frac{\sum_{i=1}^{n} (W/L)_i + \sum_{j=1}^{k} (W/L)_j}{(W/L)_{Load}},$$

where (W/L) is device width-to-length ratio and the subscripts i, j, and *Load* represent the (W/L) of stacked devices, non-stacked devices, and load devices, respectively. This equation assumes that the width-to-length ratios of the stacked devices are equal.

Simply stated, the higher the gain of the driven circuit, the lower the input down levels (Node A, Fig. 3) must be to maintain a valid up level at the output of the circuit. As circuit down levels are a function of device thresholds, which in turn are a function of the effective device channel length, it is imperative that the minimum channel length used in the design be specified in the gain circuit during analysis.

Figure 5 illustrates a typical threshold-*versus*-channel-length curve. Examination of this curve reveals that the difference in device threshold for a 3.5-μm and a 4.0-μm (artwork dimensions) channel length is 95 mV. This means that the down level necessary at Node A of Fig. 3 to maintain the same output up level for both devices must be 95 mV lower for the 3.5-μm device. Figure 6 illustrates the input-output characteristic curve for an inverter circuit, with a nominal gain of 30, for various input device channel lengths. As can be seen, circuits of the same gain can have different required input levels to maintain the same valid output up level. Therefore, use of the gain parameter to ensure dc compatibility of all circuits was valid provided that the minimum channel length of the circuits was specified.

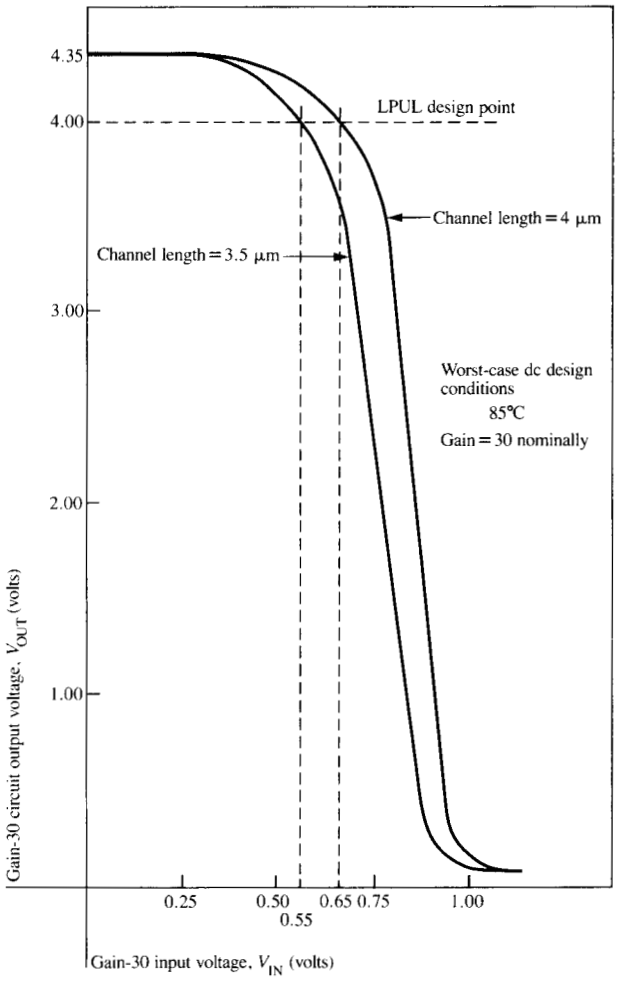

Figure 6 Gain-30 circuit input-output characteristic.

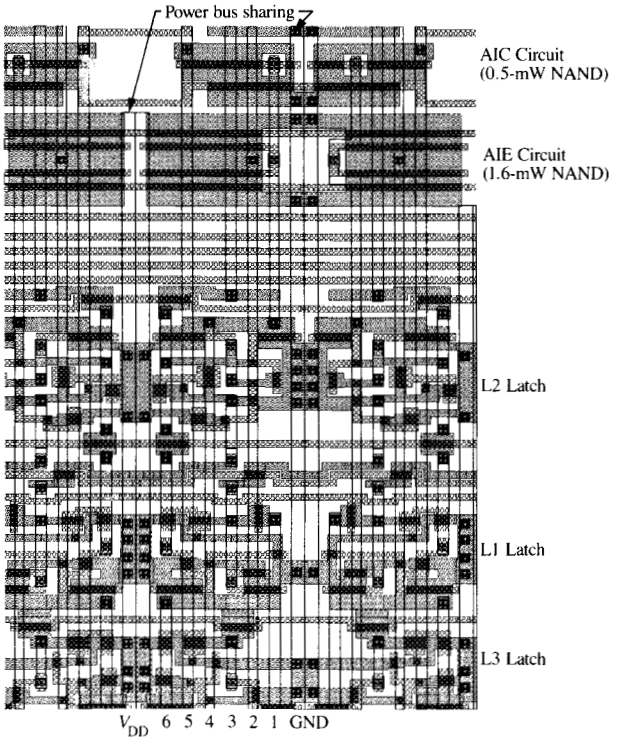

Figure 7 Typical circuit interconnectivity using variable circuit I/O.

Table 2 Best-case to worst-case performance and power spreads.

Depletion device size, artwork (μm) *(device width/ device length)*	*Performance* (ns) $(T_{on} + T_{off})/2$			*Power dissipation* (μW)		
	BC	*WC*	*WC-BC*	*BC*	*WC*	*WC-BC*
8/4	3.75	12	8.25	270	950	680
10.5/5	4.05	10.95	6.9	270	860	590
11.5/5.5	4.2	11	6.8	270	820	550

Note: T_{on} is defined as the delay for the output to fall to 1.5 volts with respect to the input attaining the 1.5-volt level. T_{off} is defined as the delay for the output to rise to 1.5 volts with respect to the input attaining the 1.5-volt level.

Since the gain parameter was independent of circuit power and logic function, a simple inverter (Stage 2, Fig. 3) was used to simulate a wide variety of driven-circuit configurations.

The dc design was accomplished by adjusting the width of device QAD (Stage 1, Fig. 3) until the output voltage of the gain-30 circuit was greater than or equal to the LPUL specification of 4.0 volts. Each random logic output stage, with the exception of the off-chip driver and special push-pull drivers, was designed to drive circuit gains of 30.

Since no random logic circuit used in the design other than an internal bus driver (which was driven by a special push-pull circuit) had a gain as high as 30, this method guaranteed a valid level for the circuits being designed. Circuits having gains higher than 30 and/or using 3.5-μm channel lengths, such as off-chip drivers, internal bus drivers, and direct input PLAs, were driven by push-pull circuits whose output down level was approximately 100 mV.

As circuit performance is a function of power dissipation, five power codes were defined for each primitive circuit. This minimized the number of circuits stored in the library and provided for simple power/performance selection. The load device sizes were specified for each power code to reduce the best-case to worst-case performance spreads due to manufacturing tolerances and to permit load device sharing between multiple circuit types of like power code.

Table 2 shows the best-case to worst-case performance and power dissipation spreads for a high-power inverter circuit (nominal power = 0.5 mW) as a function of depletion device geometry. As can be seen, a device size of 11.5/5.5 offers best-case to worst-case performance spread improvements and power dissipation improvements of 17.6% and 19.1%, respectively, when compared to a device size of 8/4. The additional area required for the larger load device size is negligible in comparison to the overall circuit size.

Basic circuits (*e.g.*, AI, OR, NOR, XOR) were combined or interconnected to create higher-level logically complex circuits. As an example, an AI and a NOR could be dotted together to form an AOI logical function. The utilization of basic circuits in this manner reduced the number of stored cells required and allowed for easy circuit expansion without further circuit design. The drain circuit wiring resistance associated with circuit dotting was accounted for in the dc design.

PLA circuits were designed with similar objectives. The input partitioning circuits were designed to be capable of driving the largest AND-array configuration, which in turn was capable of driving the largest OR-array configuration. PLA output buffers were capable of driving any logic or PLA input circuit in the library.

The design of each circuit was verified using the IBM Advanced Statistical Analysis Program (ASTAP) [7]. All process and design parameters were specified at their worst-case limits (including power supply and temperature). The value of this technique was that circuit sensitivities to various process parameters such as device lengths and widths could be accounted for in the design prior to first-pass hardware test results.

Physical layout

All random logic circuits excluding chip I/O (input/output) circuits were designed using a standard layout pitch (circuit physical width measured from the V_{DD} power supply rail to the ground rail) of 93 μm, which was extendable in 12-μm increments to 129 μm for increased wireability. The use of a standard layout pitch allowed a columnar arrangement of circuits within a macro for assembly and wiring ease.

The PLA circuit layout was accomplished in a similar manner. The input partitioning circuitry was designed using a standard layout pitch of 12.5 μm to match the AND-array pitch, while output buffers used a layout pitch compatible with that of the OR array.

Figure 7 illustrates the circuit configuration and interconnection of a typical macro. Signal wiring ran on metal, parallel to circuit metal power buses. Each circuit shared a common power supply rail with the adjacent circuit. This resulted in saved area, since separation between circuit power busing was eliminated. In addition, the circuits were designed with variable input/output locations wherever possible and with little or no metal.

The variable I/O positions permitted different circuit types to be connected together without the need of additional circuit separation for sub-metal (polysilicon, diffusion) wiring, while the lack of metal within the circuit permitted the use of metal as a circuit interconnect, resulting in efficient area utilization. Since the circuit/macro wiring ran parallel to the power busing, no crossing of power buses by signal wiring within the macro was necessary.

A typical static latch pair with push-pull output buffers is shown in Fig. 8. The outputs of the L1 latch were available on every wiring field, whereas the L2 latch output was available on every wiring field except field 6. The L1 input was limited to wiring field 1 for this configuration, whereas the secondary L2 data input was available on wiring fields 1 through 4. Pseudolevels were used to locate the various personalizations for each circuit input/output, as illustrated in Fig. 8(a). This aided the macro design process since macro designers could "trace" over these pseudolevels to obtain the desired circuit personalization without reference to other documentation.

As seen in Fig. 7, the availability of the latch output on many wiring fields aided the circuit interconnectivity. The input position of the AIE (1.6-mW NAND) circuit could be reached without the need of circuit separation for a sub-metal crossing. The AIE circuit input (channel 1) was connected to the L2 latch output via metal in channel 1. In addition, the L2 latch output was routed in channel 4 to connect to another circuit. The L1 latch output was taken

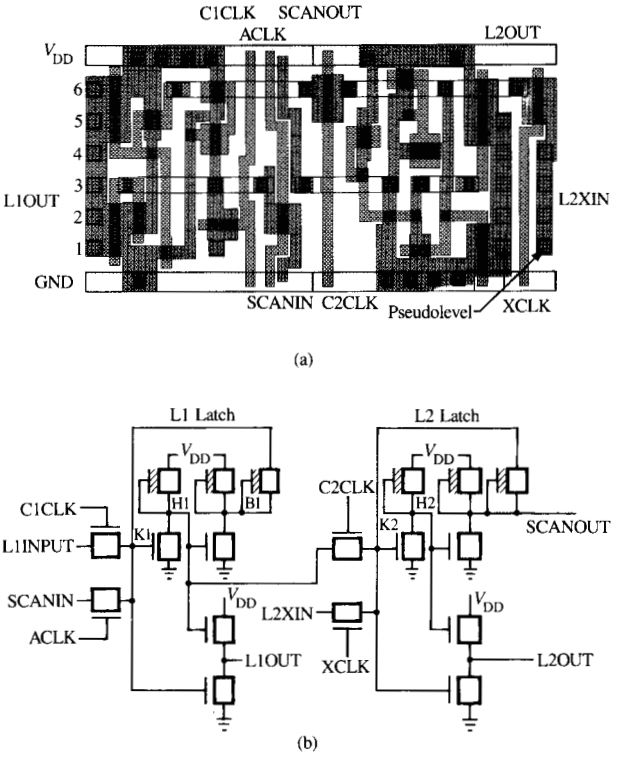

Figure 8 L1/L2 latch: (a) physical layout; (b) schematic.

from channel 6 on metal to a circuit, and also a connection was made on diffusion to the circuit directly below it. The availability of outputs on as many wiring fields as possible saved area and reduced macro assembly time. Figure 7 also illustrates the sharing of like power supply rails between adjacent circuits.

Macro design and chip design tools

All circuits and random logic macro layouts were designed using the IBM Interactive Graphics Systems (IGS) [8] and the CALMA Interactive Graphics Systems [9]. Use of these systems allowed on-line design of each circuit and eliminated the need of a pre-drafted layout for coordinate entry into the physical design data base.

Because CALMA-designed data were in a different format than the IGS format, conversion programs were written to convert the CALMA format into IBM format and vice versa. These programs allowed the design to be accomplished using both the CALMA and IGS systems concurrently, and, most important, allowed the use of IBM's extensive checking programs on CALMA design data.

The array portion of the PLAs, including personalization, was assembled automatically using a program developed

specifically for this design. The program is interactive and provisions were made for ordering inputs, product terms, and outputs without perturbing the logical data base. Use of the program, supplemented with manual folding and custom OR arrays, permitted rapid design of dense high-performance PLAs. In addition, since the layout ground rules were incorporated into the program, the resulting PLAs were free of ground rule errors. The program output was in the IGS program format and was later converted and merged with the overall chip physical design data base.

Each macro was located on the chip image to ensure wireability and that all performance requirements were met. Chip wiring or intermacro wiring was performed manually. Pre-drafted wires were entered into the physical design data base by digitizing the wiring with the CALMA system digitizer.

Checking

Since all circuits in the circuit library were stored cells which were used repeatedly in the creation of macros and ultimately the completed microprocessor, their correctness was essential. As each circuit physical layout was completed, thorough dimensional checking using the IBM Unified Shapes Checker programs (USC) [10] was performed. Any violations were corrected and the circuit rechecked. The completed and checked circuits were stored in a write-protected physical design data base to ensure that no alteration occurred inadvertently. Similarly, dimensional checking was performed upon completion of the physical layout of each macro and the intermacro global wiring.

Ensuring the correctness of each of the subassemblies prior to their incorporation to create the total chip reduced the number of errors to be corrected later in the design cycle. Logical-to-physical checking was also performed on each of the macros prior to final chip assembly. In essence, the logical description of the microprocessor was compared with the physical description for correspondence. When the program output was error free, the chip was assembled by merging each of the previously checked subassemblies together into the chip physical design data base. The entire chip was then checked as a completed entity for any errors that may have arisen from the data base merge. Only after successful completion of all checking on the final chip was the design released to manufacturing.

Results and conclusions

The entire physical design cycle of this custom 16-bit microprocessor took six months. The first-pass hardware received from manufacturing was functional. As the design was made up of nearly 23 000 transistors (6500 equivalent circuits), the potential for error was very large. A first-pass design which was functional was a major accomplishment. The processor was thoroughly tested and was sent to internal users for system development and debug. A second-pass design, which was an optimization of the first-pass design, was submitted to manufacturing four months after receiving first-pass hardware. The second-pass hardware was 100% correct. The success of the first-pass design permitted extensive analysis and testing to be completed using first-pass hardware, and this enabled the product qualification cycle to be decreased from a 20-month three-pass cycle to a 13-month two-pass cycle—a 45% reduction.

The second-pass hardware met all quality requirements and was shippable. The success of this design shows that "getting it right the first time" is indeed possible. It also indicates that a quality VLSI custom design need not cost more to produce.

In summary, the use of this hierarchical approach to physical design enabled a very complex microprocessor design to be successfully implemented in a "new" technology. The ability to design a VLSI product in such a short time reduces development cost and enables IBM microprocessor developers to provide more computing power at lower cost.

Acknowledgments

The author wishes to acknowledge the many individuals in IBM Kingston without whose efforts and support the design of the VLSI custom microprocessor would not have been possible. Specifically, the author would like to thank P. Lowy for his contributions in the area of checking methodology, G. C. Luckett and R. C. Paddock for their consultation during the preparation of this paper, and Miss D. Schussler for typing the manuscript. The author would also like to recognize E. C. Jacobson for his continued support throughout the project.

References

1. David R. Jarema and Edward H. Sussenguth, "IBM Data Communications: A Quarter Century of Evolution and Progress," *IBM J. Res. Develop.* **26,** 391–404 (1981).
2. A. J. Blodgett, "A Multilayer Ceramic Multichip Module," *IEEE Trans. Components, Hybrids, Manuf. Technol.* **CHMT-3,** 634–637 (1980).
3. E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," *Proceedings of the 14th Design Automation Conference,* New Orleans, LA, 1977, pp. 462–468.
4. R. L. Golden, P. A. Latus, and P. Lowy, "Design Automation and the Programmable Logic Array Macro," *IBM J. Res. Develop.* **24,** 23–31 (1980).
5. M. Morris Mano, *Computer Logic Design,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972.
6. H. Fleisher and L. I. Maissel, "An Introduction to Array Logic," *IBM J. Res. Develop.* **19,** 98–109 (1975).
7. *Advanced Statistical Analysis Program (ASTAP),* Program Reference Manual, Order No. SH20-1118-0; available through IBM branch offices.

8. P. Carmody, A. Barone, J. Morrell, A. Weiner, and J. Hennesy, "An Interactive Graphics System for Custom Design," *Proceedings of the 17th Design Automation Conference,* Minneapolis, MN, 1980, pp. 430–489.
9. CALMA Interactive Graphics Systems, 527 Lakeside Drive, Sunnyvale, CA 94086.
10. C. R. McCaw, "Unified Shapes Checker—A Checking Tool for LSI," *Proceedings of the 16th Design Automation Conference,* San Diego, CA, June 1979, pp. 81–87.

Received September 9, 1981; revised January 29, 1982

The author is located at the IBM System Products Division laboratory, Neighborhood Road, Kingston, New York 12401.