# Data compression technology in ASIC cores

by S. H. Burroughs
T. R. Lattrell

**IBM has made its data compression technology available to the industry through the ASIC (application-specific integrated circuit) Blue Logic core program. The papers in this journal describe four of IBM's data compression cores which are in that core library: ALDC (adaptive lossless data compression), JBIG–ABIC (Joint Bi-level Image Group–Adaptive Bi-level Image Compression), MPEG-2 (Moving Picture Experts Group-2), and 401DEC (decompression for the IBM PowerPC 401™ embedded processor). This paper is organized into three main sections: a description of the data types covered by the technology; a presentation of data compression availability through the core library elements; and a brief overview of the papers which follow.**

## Introduction

An ASIC core is defined as a function which has been designed and verified as a standalone entity and placed in the ASIC function library. The library contains the set of predesigned, pre-optimized, and pretested functions available in an ASIC (for example, 0.25-$\mu$m) technology. As such, the data compression cores in the ASIC function library represent design elements whose performance and function are guaranteed. They have been designed to core guidelines which support integration into an ASIC chip design. This has been accomplished by partitioning the functions into reusable elements or building blocks.

Since customers require the ability to tailor functions to particular applications, different types of cores were used to provide the right level of flexibility, consistent with the core-plus-ASIC design methodology. The IBM core library contains three types of cores [1, 2], which are described as *hard*, *firm*, and *soft*. A hard core is one which has a fixed physical layout and is incorporated into the design as though it were a standard cell library element. A firm core is similarly provided to the user as a library element, but the layout of the core is performed by the ASIC vendor to meet the customer layout requirements. A soft core is one in which the function is provided in the form of a technology-dependent gate-level netlist.

The importance of data compression technology in current and future systems has led IBM to identify this technology as one of the key library building blocks. Data compression methods are complex; therefore, having proven data compression algorithms implemented and available in the core library provides leverage to designers. It removes the need to be a data compression specialist or expert in order to benefit from the use of the technology. The technology was first available from IBM in standard product form, i.e., discrete devices. Therefore, to support ease of use for designers migrating from standard product offerings to ASIC chip designs, similar logic interfaces were maintained. Additional support to the ASIC designer is provided in the form of on-chip standard bus interface core library units, which minimize the chip-level core integration effort required by the designer. In terms of design migration from one ASIC technology to a successor, the designer is provided with library functions



0018-8646/98/$5.00 © 1998 IBM

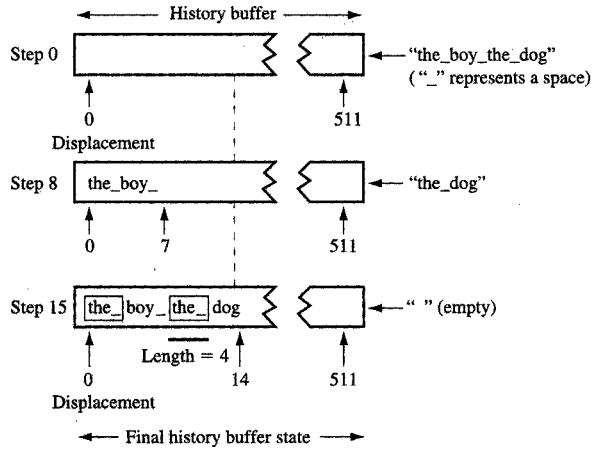


**Figure 1**

ALDC history buffer structure.

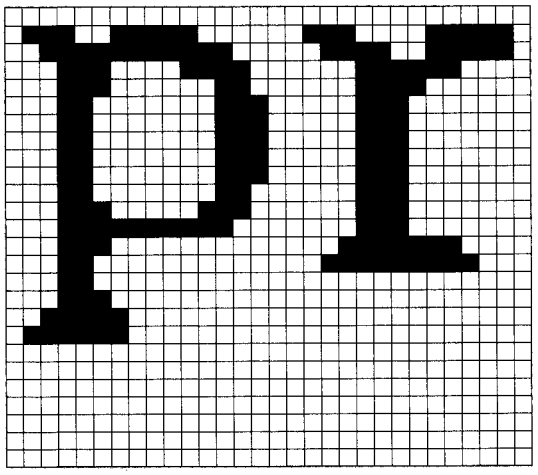

**Figure 2**

Scanned characters from a facsimile document.

which are also migrated, thereby providing a path to improved system-level performance.

## Data compression technology

Data compression has grown from a little-known technology requiring specialized training and skills to become a process that is used daily by literally millions of people. All of us who browse the World Wide Web or use the Web to download images experience this technology. Simply stated, data compression can be defined as the process of removing redundancy within text or graphic data. Eliminating such redundancy within data thus conserves the amount of space or memory required to store that information. When there is less data to transmit, the required bandwidth is minimized for such data transmissions, and thus the time required to transmit data within or across a system is minimized as well. The result is improved system efficiency.

To realize this efficiency, the compression method or technique to be applied to specific data should be based on two factors: the data type, for the most efficient compression, and the need for total recovery of the original data. A way of determining the need for total data recovery would be to compress the data and then uncompress the compressed result to determine the acceptability of that operation. To understand and know how acceptable it is if the resultant uncompressed output does not match the original data exactly is critical to selecting the right algorithm. A fully reversible (lossless) operation can be quite different from a good approximation (lossy) of the original input data.

Only lossless coding is appropriate for symbolic data, since even slight modifications to alphanumeric data would provide a result which was incorrect or unintelligible. A common lossless technique for coding symbolic data is based on work by Lempel and Ziv [3, 4] and is commonly referred to as LZ1. The LZ1 method saves a fixed amount of the most recently coded/decoded data in a history buffer. Each byte to be processed is either flagged as a "copy byte" or encoded by pointing into the history buffer and finding identical strings of bytes in the history buffer which match the input data. A length field indicates the number of characters in the past and current strings that match. This method works well for information stored in bytes with significant repetition, such as alphanumeric text or palettized graphics data. **Figure 1** shows a string of characters with pointers to string matches. This is being illustrated with an IBM ALDC (LZ1) history buffer. Because of the operational nature of this algorithm, LZ1 is described as a one-dimensional technique for byte-oriented or symbolic character data.

Facsimile images have traditionally been losslessly coded. For these black/white images, a bit stores which of two tones best represents a picture element (pel). **Figure 2** illustrates a few characters from a scanned document. The pels which make up the rectangular grid are represented in the computer as 0 (white) and 1 (black) bits. The bits are commonly packed into bytes for storage. LZ1 coding can efficiently compress large white spaces that appear as long strings of zero (0x00) bytes. However, the runs of individual white and black pels that make up the characters are not particularly on byte boundaries. Bilevel image compression techniques that consider surrounding

pels in both the horizontal and vertical directions usually achieve significantly better compression. Such techniques are considered two-dimensional compression techniques.

Two related bilevel image compression standards are the ABIC (adaptive bilevel image compression) algorithm, standardized for storing check images for the finance industry [5], and JBIG (Joint Bi-level Image Group) [6, 7]. These both look at nearby neighboring pels on the current and previous line(s) to predict the tone of the current pel. They both use adaptive probability-estimation techniques to efficiently code the prediction. These compression techniques can both be extended to graphics images with more than one bit per pel. However, the image must be decomposed into bit planes before compression.

Continuous-tone images have sufficient bits per sample to appear like pictures. JPEG (Joint Photographic Experts Group) [8, 9] compression is often used to code still images for the Internet. Although JPEG product hardware was available soon after the JPEG standard was created and it was utilized in the capture of video images and in digital cameras, most applications found JPEG software compression and decompression adequate.

For video images, the pictures are often closely related images. Significant additional compression can be achieved with MPEG** (Moving Picture Experts Group) international standards that take advantage of frame-to-frame correlations. A sequence of video frames is shown in **Figure 3**. Frames labeled I are independently decodable; frames labeled P are predicted from an earlier-in-time frame (I- or P-frame). The arrow at the top of the figure illustrates the time order of the frames. Frames labeled B are predicted from frames existing both earlier and later in time.

The first MPEG compression standard, commonly known as MPEG-1 [4], was targeted at interactive CD-ROMs. The follow-on standard, MPEG-2 [4], was intended for broadcast TV, high-definition TV (HDTV), and movies on CD-ROMs. High-speed hardware was necessary to the adoption of this technique.

Having discussed data compression for symbolic characters, fax images, continuous-tone pictures, and video images, we move on to one more special application of compression of PowerPC* instructions. Compression of the PowerPC object code reduces the cache storage requirements for such embedded processor applications. The compression can be done in software, but the decompression must be able to keep up with the processor requirements. This performance requirement makes it a necessary function for the ASIC core library.

IBM's investment in data compression technology to support its business interests in storage, processor, image, and communications markets has enabled data compression technology to evolve into one of our base

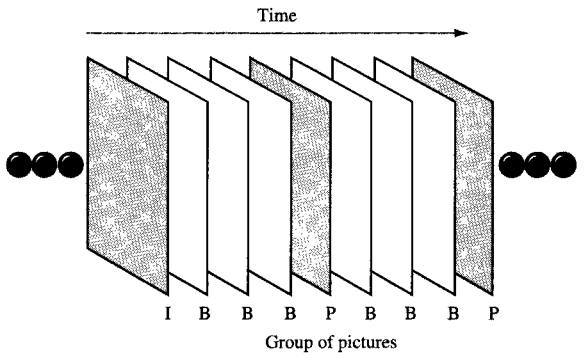


**Figure 3**

A typical group of MPEG-2 pictures in display order.

technologies in the ASIC core program. The need to exchange compressed data has driven efforts to standardize formats for compressed data. Within the company we have long been a proponent of standardization efforts and have encouraged development in this important technology.

## IBM ASIC cores

As pointed out earlier, to make something smaller than its original size is natural; it is a concept which has been prevalent throughout history. The use of symbols, pictures, and other schemes to reduce the amount of information we process is familiar in our daily lives. In the computer industry, data compression is critical to the improvement of system throughput storage and bandwidth utilization. This technology has become an enabler for several data processing applications. For example, data compression was applied to the storage industry, where it was used to compress files and databases. After early success in the storage area, advances were made in telecommunications, facsimile, and image processing. In the telecommunications industry, performance problems drove compression to solve bandwidth problems at reasonable costs. In the 1980s, fax data compression standards and digital technology made digital fax possible over analog telephone lines. The emerging application for multifunction peripheral devices having the ability to process complex images formerly handled by large host-based systems has been enabled through this technology. By utilizing standardized data compression algorithms, such as JBIG or ABIC, image-intensive high-quality desktop publishing tasks can be performed on relatively low-cost hardware. The hardware implementation of these algorithms can be a cost-effective solution to system throughput, bandwidth, and storage utilization. The rest of this section discusses

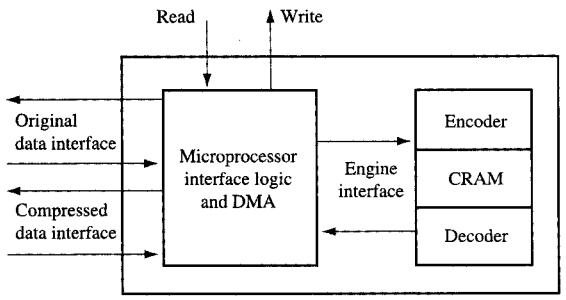


Figure 4

ALDC block diagram.

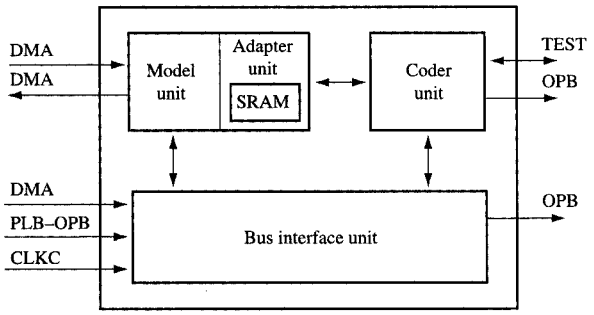


Figure 5

JBIG–ABIC firm core.

briefly the high-level architectural block diagrams for the four IBM ASIC data compression cores.

• *ALDC—adaptive lossless data compression*
The ALDC function has been partitioned into two elements: a soft core, containing peripheral interface logic, and a hard core, containing the encode/decode function, as shown in **Figure 4**. The hard core has the timing-critical elements of the design: the history buffer and the encode/decode engine. The soft core provides a user-friendly interface to the engine, data flow control, and some buffering. It resembles the interface of IBM's ALDC standard products. The throughput of the ALDC core is 100 MB/s, processing one byte per cycle. This core supports design migration to ASICs for the system-level designer and also provides a substantial performance improvement over the standard product offerings.

The data history function has been designed using a proprietary [10, 11] high-speed CAM (content-addressable memory) technology referred to as a CRAM (content-addressable random-access memory). Since the history size is application-dependent, hardware instantiations with 512, 1K, and 2K bytes have been implemented as separate individual cores. This permits customer selection of the configuration which best meets the application requirements.

• *JBIG–ABIC—lossless bilevel image compression*
The JBIG–ABIC core provides a hardware assist to the system designer which reduces CPU (processor) overhead during data compression for bilevel images. The throughput objective is to process one bit per cycle; therefore, with a clock rate of 80 MHz, the throughput would approach 80 Mb (pels) per second. The JBIG–ABIC function has been implemented as a firm core. It contains three subblocks in the design, as shown in **Figure 5**: the model and adapter unit (with embedded SRAM), the coder unit, and the bus interface unit. The core was designed to provide configurability through a programming interface. This implementation supports three modes of operation: two standard modes and a hybrid combination of the two. The two standards which are supported are ABIC, as used in the banking industry, and ITU-T Rec. T.82 | ISO/IEC 11544 (JBIG) [7], as used in the fax industry. Various parameters can be set to further control the operation and tailor the function to application requirements.

• *MPEG-2—lossy video image data compression*
The MPEG-2 function has been partitioned into a set of three cores which combine to form the subsystem shown in **Figure 6**. The transport demultiplexor unit, a soft core, parses the input data into audio, video, and system data. The video decoder is a firm core which decompresses the MPEG-2 video data stream and passes it to a digital video device. The audio decoder is a soft core which decompresses the audio data and provides the uncompressed digital audio data to an external digital-to-analog converter. Each of these cores can be used independently, but additional advanced features are enabled when they are used together. For example, complex multicore audio/video operations such as channel changes, time changes, synchronization, and error concealment are all done in hardware, minimizing the impact on software and further simplifying product development and enhancing the end-user product.

• *401DEC—PowerPC 401* instruction code decompression*
Code size efficiency is an important consideration in embedded system design. The 401DEC core was designed to provide the ability to decompress PowerPC instructions "on the fly" at full processor speeds from a code space in

which compressed instructions have been located. This soft core is a single functional entity which has all the required bus interfaces to support its use with the PowerPC 401 embedded processor core (**Figure 7**).

## Papers in this special issue

All of the algorithms described in the collection of papers in this special issue on the IBM ASIC data compression cores have been implemented in silicon. The ALDC paper discusses our lossless character compression core, which was first offered in standard product form. This algorithm was adopted as a QIC standard and has been shipped in 5/20/40MB/s ALDC standard products for several years. The core implementation has a performance of 80–100 MB/s. More recently, the JBIG–ABIC core, at a performance rate of 80 MHz, was added to the library and has been shipped since early 1997. The MPEG-2 subsystem is a set of three cores which were added to the library in the fourth quarter of 1997. MPEG-2 was first offered in standard product form before being added to the core library. Finally, a paper describing an instruction decompression core for the PowerPC is presented. This function has been available since mid-1997. More details on the papers are given next.

*A fast hardware data compression algorithm and some algorithmic extensions*

This paper presents exciting results for a family of efficient hardware compression engines with very high performance which are developed for a CAM (content-addressable memory) hardware base [11]. An overview of the hardware and encoding structure for ALDC and for the BLDC extensions is presented with tabulated results.

*Design considerations for the ALDC cores*

Textual data may be organized as single blocks of data, normally referred to as files, or as multiple blocks of data, called segmented data. The ALDC core [10] provides support for segmented data. The core is capable of processing both blocked and unblocked data. This paper describes the architectural features of the core, which incorporate automatic segmentation and history buffer controls to work with a segmented data type.

*A JBIG–ABIC compression engine for digital document processing*

This paper describes a core which was completed for the multifunction peripheral (MFP) market through an alliance between IBM Microelectronics and Xionics Document Technology Inc. The problems faced in the MFP arena and the architectural features incorporated into the JBIG–ABIC core [12], which provide a hardware solution to this industry, are described in the paper.

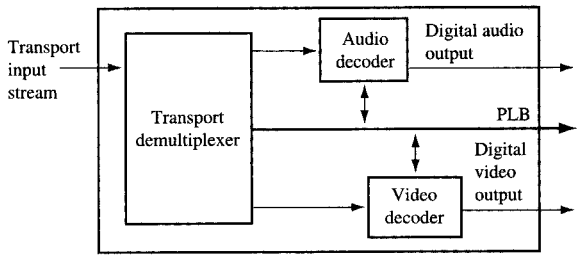


Figure 6

MPEG-2 subsystem.

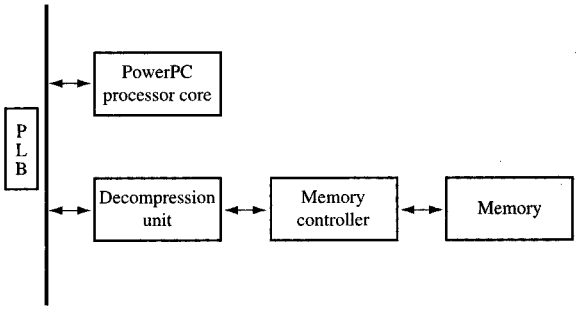


Figure 7

Embedded PowerPC configuration with decompression.

*Performance as a function of compression*

A performance review of the ABIC, JBIG ALDC, and BLDC data compression algorithms contained in the core library is presented, with a focus on the critical parameters [13] of throughput and latency. This presentation of lossless compression algorithms offers an exciting insight into the compressibility of image data and the throughput performance.

*The Qx-coder*

An in-depth review [14] of the hardware-optimized Q-coder contained in the ABIC algorithm and the software-optimized QM-coder as incorporated in JBIG is presented. This study explores the differences between the two coders, and a proposed solution demonstrates the ability to merge Q-coder and QM-coder functions in shared logic, called the Qx-coder. The technique to integrate both coders to produce the Qx-coder is novel and has resulted in a major contribution to the availability of a merged hardware solution.

*JBIG–ABIC macro algorithm verification*
A description of the JBIG–ABIC Verification Suite [15] is presented. The IBM suite of images is shown to go well beyond the set of standard images which was available in the industry. The contents of the suite are described; an example is then presented for a compression subsystem based on the JBIG–ABIC core.

*Integrating the MPEG-2 subsystem for digital television*
This paper addresses the integrated subsystem consisting of the transport demultiplexor, audio decoder, video decoder [16], and supporting host processor and memory subsystem elements. The dataflow through the subsystem is presented from an application viewpoint. The advantages of the core implementation are clearly shown, and the reduction in system complexity is highlighted. The paper describes the advantage of standardized on-chip bus interface units to support processor connectivity and shared memory requirements; corresponding savings at the system level translate to reduced gate count and performance improvements.

*A decompression core for PowerPC*
This paper presents a data compression method [17] which was developed to improve code size efficiency through the use of compression techniques applied to code storage. The resultant decompression core enables an instruction decode "on the fly" for the PowerPC 401 embedded processor. The implementation details are developed and results shown for various compilers.

## Summary

The IBM Microelectronics Division is committed to maintaining a leadership position in the industry as a solutions provider through the functions in the ASIC core program. The data compression cores presented in this journal in the accompanying papers are key components.

## Acknowledgments

The authors wish to express their appreciation to Dr. Joan Mitchell for her dedicated assistance in the JBIG–ABIC core development and verification project and for her continued consulting on all aspects of data compression. Without her assistance and dedication this issue of the *Journal* would not have been possible. The authors would also like to recognize and thank Dr. Ron Arps of the Almaden Research facility for his contributions to data compression technology; his leadership and excellent record of research in image data compression have contributed greatly to the business, especially in the development of the ABIC standard. The authors also acknowledge Lili Udell for her assistance with the figures used in this paper.



## References

1. Ann Marie Rincon, Cory Cherichetti, David R. Stauffer, and Michael Trick, "Core+ASIC Methodology: The Pursuit of System-on-a-Chip," presented at Wescon IC Expo '97, Santa Clara–San Jose, CA, November 4–6, 1997.
2. Ann Marie Rincon, Michael Trick, and Thomas Guzowski, "A Proven Methodology for Designing One-Million Gate ASICs," *Proceedings of the IEEE Custom Integrated Circuits Conference*, May 1997, pp. 45–52.
3. J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. Info. Theory* **IT-23**, No. 3, 337–343 (1977).
4. Roy Hoffman, *Data Compression in Digital Systems*, Chapman & Hall, New York, 1997.
5. R. B. Arps, T. K. Truong, D. J. Lu, R. C. Pasco, and T. D. Friedman, "A Multi-Purpose VLSI Chip for Adaptive Data Compression of Bilevel Images," *IBM J. Res. Develop.* **32,** 775–795 (1988).
6. W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, Jr., and R. B. Arps, "An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder," *IBM J. Res. Develop.* **32,** 717–726 (1988).
7. *ITU-T Rec. T.82 | ISO/IEC 11544:1993 Information Technology*—Coded Representation of Picture and Audio Information—Progressive Bi-Level Image Compression (JBIG standard).
8. *ITU-T Rec. T.81 | ISO/IEC 10918-1:1993 Information Technology*—Coded Representation of Picture and Audio Information—Digital Compression and Coding of Continuous-Tone Still Images (JPEG standard).
9. W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993 (ISBN 0-442-01272-1).
10. M. J. Slattery and F. A. Kampf, "Design Considerations for the ALDC Cores," *IBM J. Res. Develop.* **42,** 747–752 (1998, this issue).
11. D. J. Craft, "A Fast Hardware Data Compression Algorithm and Some Algorithmic Extensions," *IBM J. Res. Develop.* **42,** 733–745 (1998, this issue).
12. K. M. Marks, "A JBIG–ABIC Compression Engine for Digital Document Processing," *IBM J. Res. Develop.* **42,** 753–758 (1998, this issue).
13. F. A. Kampf, "Performance as a Function of Compression," *IBM J. Res. Develop.* **42,** 759–766 (1998, this issue).
14. M. J. Slattery and J. L. Mitchell, "The Qx-Coder," *IBM J. Res. Develop.* **42,** 767–784 (1998, this issue).
15. P. S. Colyer and J. L. Mitchell, "The IBM JBIG–ABIC Verification Suite," *IBM J. Res. Develop.* **42,** 785–794 (1998, this issue).
16. R. E. Anderson, E. M. Foster, D. E. Franklin, and R. S. Svec, "Integrating the MPEG-2 Subsystem for Digital Television," *IBM J. Res. Develop.* **42,** 795–805 (1998, this issue).
17. T. M. Kemp, R. K. Montoye, J. D. Harper, J. D. Palmer, and D. J. Auerbach, "A Decompression Core for PowerPC," *IBM J. Res. Develop.* **42,** 807–812 (1998, this issue).

*Received May 28, 1998; accepted for publication September 8, 1998*

**Stuart H. Burroughs** *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (sburroug@us.ibm.com).* Mr. Burroughs joined IBM in 1968. Since that time he has had numerous technical and management assignments. His current assignment is as a Program Manager in the IBM Core Plus ASIC development program. As program manager he oversees the development and acquisition of reusable logic functions. Recent and current projects include embedded microprocessor logic such as picoJava. He has been heavily involved in the development of strategic alliances with several companies for the Core Plus ASIC area. Prior to his current position in the development of ASIC cores, Mr. Burroughs's recent assignments include design manager, technical applications, and product marketing manager for the IBM Palette DAC products which were introduced to the merchant market. In the image compression market, he was a principal in the business development of the JBIG–ABIC core. During his career at IBM, Mr. Burroughs has been instrumental in the development of standard products using lossless data compression, palette DAC graphics logic products, and compression cores. He has also made contributions in the application of finite-element-method modeling techniques for electronic packaging. He received both bachelor's and master's degrees in mechanical engineering from the University of Vermont. Over his professional career, he has published numerous papers and technical reports in technical journals and trade publications. He is a Member of ASME International.

**Ted R. Lattrell** *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (tlattrell@us.ibm.com).* As Program Manager of IBM's Blue Logic core program, Mr. Lattrell guides the development of IBM's portfolio of intellectual property for the program. In his 17 years at IBM, he has accumulated invaluable experience in technology, logic, and design methodology development. The success of IBM's Blue Logic core program is in part due to his experience and effort in integrating these interdependent disciplines. Prior to his current assignment with IBM Microelectronics, Mr. Lattrell was instrumental in IBM's development of adaptive lossless data compression (ALDC). A part of his responsibility was to work as a liaison with international standards committees for the adoption of ALDC. Today, ALDC is recognized as a QIC standard and is under review for ANSI certification. In addition, Mr. Lattrell's work with the development of IBM's first VHDL design methodology, including simulation, synthesis, and timing tools, helped to reduce the time required to design integrated circuit logic. His foundation of work with IBM is grounded in the development of metal-on-oxide technologies. He assisted in the qualification of the n-MOS and original CMOS technologies and later led the qualification of IBM's 1.5-$\mu$m CMOS technology.

[[Page 732 is blank]]