The IBM JBIG–ABIC Verification Suite

by P. S. Colyer
J. L. Mitchell

The IBM JBIG–ABIC Verification Suite is a newly designed verification suite that contains more than 5000 correctly encoded test images. Previously existing verification images in total and consolidated in an *ad hoc* suite tested only a small fraction of the algorithm and were inadequate. The IBM JBIG–ABIC Verification Suite provides compatibility testing reference data for the ITU-T/ISO JBIG sequential mode and IBM ABIC image compression standards. In this paper, the test images are described and related to the JBIG and ABIC compression standards and options. The verification suite was used to debug and verify the algorithms in the IBM JBIG–ABIC core that is integrated into the Xionics XipChip imaging microcontrollers.

Introduction

JBIG is the Joint Bi-level Image Experts Group [1] image compression standard, and ABIC is the IBM Adaptive Bilevel Image Compression [2] standard. The JBIG and the ABIC image compression standards are reversible and lossless. Therefore, each correctly decoded image is identical to the corresponding original source image. Also, each correctly encoded test image will be identical to the reference encoded image in the verification suite.

Correct implementation of the JBIG and ABIC algorithms is critical. Garbled, unrecognizable, and possibly uncorrectable reconstructed images result from incorrect encoding or incorrect decoding. If the algorithm implementation has an error, and the original source image is unchanged, attempts to re-encode the source image would also fail. Also, if the original source image is not available, an encoding error could result in lost data, since the decoding algorithm requires correctly encoded data.

In combination, the IBM JBIG–ABIC Verification Suite is a collection of new reference data containing about 5000 encoded images and the corresponding source images. The verification suite is useful for testing implementations of the JBIG and ABIC compression algorithms.

Before the development of the verification suite, the readily available standard images consisted of six JBIG sequential-mode images, seven JBIG progressive-mode images [3], and no ABIC images.

The new IBM image verification suite contains source images, JBIG sequential-mode encoded images, and ABIC encoded images. To provide a range of tests for the various coding options and algorithm features, the source images were encoded with various compression parameters, and the resulting encoded images are included as part of the verification suite. The encoded images and source images are grouped in five general categories:

- Publicly available images.
- Custom-targeted images.
- Parameter and marker variations.

0018-8646/98/$5.00 © 1998 IBM

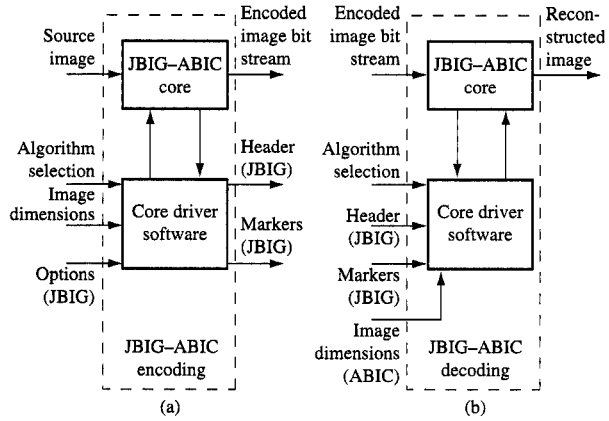

Figure 1

Implementation of JBIG–ABIC core: (a) encoding; (b) decoding.

- Randomly generated images.
- Implementation-specific images.

The verification suite will spare the user many hours of devising test strategies and creating test cases, as well as the need to learn and understand how to use finite-precision mathematics to represent infinite-precision fractions as very large numbers, as is required for effective test cases. Nevertheless, a thorough verification process will require understanding the specifications, learning the algorithm details, learning arithmetic coding, and devising additional testing strategies. The verification suite provides tests for a significant portion of the more difficult to test encoding and decoding considerations such as model template states, renormalization, probability-estimation table traversal, image boundaries, AT moves, and register flushing.

For maximum test coverage and minimum test time, 586 of the source images were systematically designed; each of them contains only a small number of lines or less of image data, and is a fraction of the size of a typical, real-world image. The small source images are also referred to as "bit sequences" to emphasize the significance of the sequence of the bits.

The verification suite was used to verify that the JBIG and ABIC algorithms were correctly implemented in the IBM JBIG–ABIC core [4] that is integrated into the Xionics XipChip imaging microcontrollers. It is important to note that the IBM JBIG–ABIC core design includes the Qx-coder [5], which is the merger of the QM-coder (JBIG) [1, 6, 7] and the Q-coder (ABIC) [1, 8, 9]. The verification suite was encoded and decoded by a simulation model of the IBM JBIG–ABIC core. The simulation results were identical to the correct reference results included in the verification suite.

The IBM JBIG–ABIC Verification Suite was not designed to be a universal test to verify all possible cases; the suite is not intended, and is not sufficient, to test implementation-specific particulars such as pipelines, state machines, and timings. Also, the particulars of algorithm implementation may introduce additional verification considerations that are not tested by the verification suite. Implementation-specific particulars often include design decisions, optimization choices, and system-level considerations that are not defined in the JBIG and ABIC specifications.

Verification of the IBM JBIG–ABIC core

The IBM JBIG–ABIC Verification Suite has application in the verification of coding algorithms used in compression subsystems. What follows is a brief description of JBIG and ABIC image encoding and decoding by an example compression subsystem based on the IBM JBIG–ABIC core. The core design provides compression-services hardware assistance to the system processor.

Figure 1 shows the example JBIG–ABIC implementation. Dashed boxes represent the compression subsystem. The two subblocks represent the core driver software and the IBM JBIG–ABIC core.

The core driver software uses image dimensions, header parameters, and markers to program and monitor the core. The core driver software also generates the headers and inserts markers in the encoded image bit stream. The IBM JBIG–ABIC core encodes and decodes the input data according to the selected algorithm and coding options.

- *Encoding*

In the example subsystem configured for encoding shown in Figure 1(a), the inputs for encoding are the source image, algorithm selection, options (JBIG), and image dimensions. The algorithm selection input is used to choose the coding algorithm (JBIG or ABIC).

Encoding by the JBIG algorithm requires the input bit-stream representation of the source image, the image horizontal dimension, the image vertical dimension, and the coding options. The JBIG algorithm encoded output is a JBIG header and an encoded image bit stream. The header specifies the parameter settings, image dimensions, and coding options. The encoded data output is the encoded image bit stream with JBIG markers inserted at each stripe boundary.

Encoding by the ABIC algorithm requires only the input bit-stream representation of the source image and two parameters: 1) the image horizontal dimension and 2) the image vertical dimension. The ABIC algorithm encoded

Table 1 JBIG parameters.

Parameter	*Name*	*Size* (bits)	*JBIG range*	*Core range*
DL	Number for initial resolution layer	8	0–255	0
D	Number for final resolution layer	8	0–255	0
P	Number of bit planes	8	1–255	1–255 (SW*)
Reserved	Reserved	8	0	0
Xd	Horizontal dimension (number of pixels per line)	32	1–4,294,967,295	1–65,530
Yd	Vertical dimension (number of scan lines)	32	1–4,294,967,295	1–65,530 × Nstripes[†]
L0	Lines per stripe at layer 0	32	1–4,294,967,295	1–65,530
Mx	Maximum horizontal adaptive template pixel move	8	0–127	0, 3–127
My	Maximum vertical adaptive template pixel move	8	0–255	0
Reserved	Reserved	4	0	0
HITOLO	High to low (plane ordering)	1	0–1	0–1 (SW[1])
SEQ	Sequential order	1	0–1	0–1 (SW[1])
ILEAVE	Interleave multiple bit planes	1	0–1	0–1 (SW[1])
SMID	Index over stripe is in middle	1	0–1	0–1 (SW[1])
Reserved	Reserved	1	0	0
LRLTWO	Lowest resolution layer two-line template	1	0–1	0–1
VLENGTH	Variable-length marker allowed	1	0–1	0–1 (SW[1])
TPDON	Differential layer typical prediction enabled	1	0–1	0
TPBON	Lowest resolution layer typical prediction enabled	1	0–1	0–1
DPON	Deterministic prediction enabled	1	0–1	0
DPPRIV	Deterministic prediction private table	1	0–1	0
DPLAST	Deterministic prediction last table reused	1	0–1	0
DPTABLE	Deterministic prediction table	0 or 1728[‡]	–	–

*SW denotes that functional support requires driver software interaction with the core.
[†]The maximum Yd for the core is 65530 × the number of stripes in the image.
[‡]In sequential mode, the DPTABLE size is 0.

data output is the compressed-image bit stream. Note that in complete system implementations, the ABIC algorithm compressed images are usually embedded within an IOCA (Image Object Content Architecture) [10] header that contains the image dimension parameters. The ABIC algorithm itself specifies no header; the IOCA standard defines a header for use with ABIC.

The coding algorithms can be verified by encoding source images having various image data characteristics with various coding parameters. Significantly, correctly encoded data generated by the core exactly match the encoded data included in the verification suite.

- *Decoding*

The example subsystem configured for decoding is shown in Figure 1(b). The inputs for decoding are encoded image bit stream, algorithm selection, header (JBIG), markers (JBIG), and image dimensions (ABIC). The algorithm selection input is used to choose the coding algorithm (JBIG or ABIC). The decoding output is the reconstructed source image.

Decoding JBIG encoded images requires the JBIG header and the encoded image bit stream with the JBIG markers included. Decoding ABIC encoded images requires the image dimensions and the encoded image bit stream.

The decoding algorithms can be verified by decoding encoded data having various data characteristics that have been encoded with various coding parameters. Again, correctly decoded data generated by the core exactly match the source images included in the verification suite.

Algorithm parameters and control markers

The JBIG algorithm has optional features that are selected and controlled by parameters and markers. The ABIC algorithm has no options and requires only two parameters. All of the parameters and markers are briefly described below.

- *JBIG parameters*

The JBIG algorithm is controlled by 20 parameters. The JBIG parameters are set before encoding begins, and the parameters are stored in the JBIG header for use during decoding. The JBIG header precedes the encoded data in the finished JBIG encoded image file.

In JBIG sequential mode, one of the parameters is not used, and five of the parameters are always set to 0. (Options required only for progressive-mode coding are not varied and are set to fixed values in the verification suite.) The verification suite has no new images for JBIG progressive mode. The few publicly available progressive-mode images are included in the verification suite.

Table 2 JBIG markers.

Marker	*Name*	*Segment size* (bytes)	*JBIG value*	*Core use*
RESERVE	Reserve	2,3,. . .*	0xFF01	No
SDNORM	Stripe end data normal	2	0xFF02	Yes
SDRST	Stripe end data reset	2	0xFF03	Yes
ABORT	Abort	2	0xFF04	Yes
NEWLEN	New vertical dimension (Yd)	6	0xFF05	Yes (SW†)
ATMOVE	Adaptive template movement (YAT, tx, ty)‡	8	0xFF06	YAT (SW), tx, ty = 0
COMMENT	Private comment	6 + n§	0xFF07	Yes (SW†)
START-JBIG-IMAGE	T.85 fax image start	2	0xFFA8	Yes (SW†)
END-JBIG-IMAGE	T.85 fax image end	2	0xFFA9	Yes (SW†)

*The number of bytes used by the RESERVE marker segment is not specified.
†SW denotes that functional support requires driver software interaction with the core.
‡At the YAT scan line of the next stripe, move the adaptive template (AT) pixel to position (tx, ty).
§Where n is number of bytes used to represent the comment.

Table 1 shows each JBIG parameter mnemonic, parameter description, and number of header bits used to specify the parameter value, with the range of parameter values allowable in the JBIG algorithm and the range of values implemented in the IBM JBIG–ABIC core. The parameters are listed in the table in the order in which they appear in the JBIG header.

The initial resolution layer (DL) and the final resolution layer (D) are always 0 in sequential mode. The precision (P) is the number of bit planes. P is 1 for bilevel images (i.e., black and white). Since each bit plane is independently encoded, the core can be used for encoding multiple-bit-per-pixel images under control of the driver software. In the JBIG header, the precision (P) is followed by eight reserved bits that must be set to 0.

The image horizontal dimension (Xd) and the image vertical dimension (Yd) specify the number of pixels per line and the number of lines in the image. The lines are grouped into stripes composed of L0 lines per stripe at layer 0. In the IBM JBIG–ABIC core, Xd and L0 are limited to 65,530. The maximum vertical dimension allowed by the JBIG specification is available with the core by the use of multiple stripes to achieve the full range of vertical dimension.

The maximum horizontal and vertical displacements allowed for the adaptive template (AT) pixel are specified in the parameters Mx and My. In the core, the AT pixel is either at the default position on the first previous line, or on the current line at up to 127 pixels to the left of the pixel being encoded. The parameter My is followed by four reserved bits that must be set to 0.

The next four parameters (HITOLO, SEQ, ILEAVE, and SMID) specify the order of the stripes in the encoded data. In the IBM JBIG–ABIC core, the stripe order is handled by the core driver software. The stripe order has no effect on the encoding and decoding algorithms. The parameter SMID is followed by one reserved bit that must be set to 0.

Template selection is indicated by the parameter LRLTWO. LRLTWO is 1 for the two-line template and 0 for the three-line template. The VLENGTH bit enables the optional use of the NEWLEN marker, which can be used to indicate a new, smaller image vertical dimension after decoding has begun. In the IBM JBIG–ABIC core, the core driver software handles VLENGTH. Typical prediction (TP) for the lowest resolution layer is enabled with the TPBON parameter.

The TPDON, DPON, DPRIV, and DPLAST parameters are used only in progressive-mode coding. In sequential-mode coding, TPDON, DPON, DPRIV, and DPLAST are always set to 0, and the DPTABLE parameter size is 0.

- *JBIG markers*

The nine JBIG markers are inserted in the encoded image bit stream at stripe boundaries during encoding. The markers are inserted only as needed to direct the decoding process. **Table 2** is a complete list of the JBIG markers, indicating the names, segment sizes, JBIG values, and marker usage by the IBM JBIG–ABIC core. A brief description of each JBIG marker follows.

The RESERVE marker is for private use and is not uniquely defined in the JBIG specification. The RESERVE marker is not implemented in the IBM JBIG–ABIC core.

The SDNORM marker immediately follows the encoded image bit stream for stripes for which the coder algorithm internal state is to be preserved and used to encode the next stripe. The SDRST marker immediately follows the

Table 3 ABIC parameters.

Parameter	*Name*	*ABIC range*	*Core range*
Xd	Horizontal dimension (number of pixels per line)	1–4,294,967,295	1–65,530
Yd	Vertical dimension (number of scan lines)	1–4,294,967,295	1–65,530

Table 4 Publicly available images.

Name	*Image count*	*Algorithm*	*Description*	*Purpose*
table26	1	JBIG	ISO/IEC 11544: 1993 (E) Table 26	QM-coder
img2, img3, img4n	6	JBIG	U.S. JBIG Committee X3L3.2 Juried Validation Set—Sequential Mode	Marker codes
img1, img2, img3, img4n	7	JBIG	U.S. JBIG Committee X3L3.2 Juried Validation Set—Progressive Mode	Marker codes
ptt_n	8	JBIG	CCITT Group 3 two-dimensional standardization images	Normal image dimensions
jm88a2	1	ABIC	*IBM J. Res. Develop.* **32,** 753–774 (1988)	Q-coder

encoded image bit stream for stripes for which the next stripe is to be encoded with initial conditions such that the next stripe can be decoded independently.

The ABORT marker, which signals the end of the available encoded data, is used to terminate the decoding of an image prematurely. Because the JBIG decode algorithm requires all of the encoded data to be decoded to the full image dimensions, the ABORT marker is useful when the encoded data become unavailable (i.e., a dropped transmission line, a failed retrieval system, corrupted encoded data, or other system failure).

The NEWLEN marker allows a new, smaller image vertical dimension (or length) to be set (e.g., during a fax transmission) that is less than the image vertical dimension set in the JBIG header, provided VLENGTH is set to 1. The NEWLEN marker is particularly handy for facsimile environments in which the scanned paper may be shorter than expected. When the NEWLEN marker results in early termination of the last stripe of the image, a null (dummy) stripe must follow the NEWLEN marker.

The ATMOVE marker is followed by the number of the scan line of the next stripe (YAT) before which the adaptive template (AT) pixel is to be moved to the new position specified by the AT pixel horizontal displacement (tx) and AT pixel vertical displacement (ty) parameters. Note that the JBIG specification limits tx and ty such that $tx \leqq Mx$ and $ty \leqq My$.

The COMMENT marker indicates a "private comment" that is to be ignored by the JBIG algorithm. The COMMENT marker is followed by four bytes that specify the comment length (n) in bytes.

The T.85 fax application standard [11] specifies a start of JBIG image marker (START-JBIG-IMAGE) and an end of JBIG image marker (END-JBIG-IMAGE). The fax standard adds the start and stop markers to provide confirmation that the decoder received complete data. ITU-T/ISO JBIG images do not start and stop with unique two-byte codes. The T.85 FAX standard added the start and stop markers to the ITU-T/ISO JBIG standard.

In the IBM JBIG–ABIC core, the ABORT, SDNORM, and SDRST markers are handled by the core, and the other markers are handled by the core driver software.

- *ABIC parameters*

The ABIC algorithm is controlled by two parameters: the image horizontal dimension (Xd) in pixels and the image vertical dimension (Yd) in lines. **Table 3** shows the range of ABIC algorithm compatible image dimensions, and the implemented range of ABIC image dimensions for the IBM JBIG–ABIC core. The ABIC standard allows image horizontal and vertical dimensions of 4,294,967,295. In the core, the image horizontal dimension (Xd) and image vertical dimension (Yd) are limited to 65,530.

Contents of the IBM JBIG–ABIC Verification Suite

The IBM JBIG–ABIC Verification Suite is composed of images and data that are grouped in five general categories: publicly available images, custom-targeted images, parameter and marker variations, randomly generated images, and implementation-specific images.

For all source images and encoded data in the verification suite, the source image data is in raster form, with eight

Table 5 Custom-targeted images.

Name	*Image count*	*Algorithm*	*Description*	*Purpose*
qm0. . .32	140	JBIG	Traverse QM-coder table (coder unit only)	Q-value table and adapter unit
qc0. . .13	86	ABIC	Traverse Q-coder table (coder unit only)	Q-value table and adapter unit
jamqm0. . .35	149	JBIG	Traverse QM-coder table	Q-value table and adapter unit
jamqc0. . .18	120	ABIC	Traverse Q-coder table	Q-value table and adapter unit
stack	3	JBIG	Many 0xFF bytes encoded	Stack counter
carry	3	JBIG	Many 0xFF bytes encoded with a carry	Stack counter
fax21_carry	1	JBIG	Page fax image with carry	Stack counter
fax31_stack	1	JBIG	Page fax image without carry	Stack counter
jbig1. . .jbig6	68	JBIG	All possible JBIG template states	Model unit
jbig1_jbig1_n	1	JBIG	All possible JBIG two-line template edge states	Model unit
jbig4_jbig4_n	1	JBIG	All possible JBIG three-line template edge states	Model unit
abic1	1	ABIC	All possible ABIC template states	Model unit
abic1_abic1	1	ABIC	All possible ABIC template edge states	Model unit
jbig_31	4	JBIG	Data volume expands on compression	Throughput control
trailing 00	7	JBIG	Encoded trailing 00 bytes truncated	Decode trailing 00 termination

pixels per byte. The background is assumed to be white (0) and the foreground is assumed to be black (1). The first bit of every line is located at the leftmost bit of a byte.

• *Publicly available images*

The publicly available images are listed in **Table 4**, which shows the image name, the number of test cases, the algorithm that applies, a brief description, and the main purpose of the test.

The publicly available images test only a small subset of all possible coding algorithm situations. Because of the nature of the images, some incorrect implementations of the JBIG and ABIC algorithms would correctly encode and decode the publicly available images.

The JBIG standard specification [1] includes a bit sequence listed as table26 that is encoded with an artificial context. The bit sequence table26 is useful for the QM-coder (JBIG) because the specification includes both the encoded data and the intermediate internal register results. The bit sequence is not useful for the complete JBIG algorithm because the required artificial contexts cannot be generated by a correct model unit.

The verification suite contains images from the U.S. JBIG Committee (X3L3.2) peer-reviewed JBIG validation images that are listed as img1, img2, img3, and img4n. The X3L3.2 images provide examples of encoding by both sequential mode and by progressive mode, and the X3L3.2 images make heavy use of the marker codes.

In addition, the eight images used to standardize ITU-T Rec. T.6 (MR and MMR) [12, 13] are included as PTT_n (n is 1 to 8). These images were too large for the simulation environment that was used to verify the IBM JBIG–ABIC core, but they are useful for verifying actual hardware and software implementations. The PTT images are typical full-page facsimile images.

The publicly available bit-sequence image listed in Table 4 as jm88a2 is particularly useful because the internal register results for both hardware and software conventions are listed in [8].

• *Custom-targeted images*

The custom-targeted images are listed in **Table 5**. Most of the custom-targeted images are designed to test the coder unit; the rest are designed to test the stack counter, the model unit, output latency, and encode/decode termination.

The custom-targeted images are the result of much studying of algorithm register values while encoding, modifying the bit sequence, and re-encoding until effective bit sequences were found. Also, for many cases, manual encoding and re-encoding of specific bit sequences was required to develop bit sequences that achieve the needed test effect. It is important to note that the custom-targeted images alone are not sufficient to fully test any given compression subsystem implementation.

The image names beginning with the letter q are single-line bit sequences for testing the coder unit only. The qm bit sequences are for the QM-coder (JBIG), and the qc bit sequences are used for the Q-coder (ABIC). The q bit sequences must be used with only one context for the entire pattern to achieve the desired test coverage of the Q-value table, coder unit, adapter unit, bit/byte stuffing, and the flush process. Some q patterns have a and b versions; the a versions contain predominantly 0 bits, and the b versions contain predominantly 1 bits. Using the a and b versions provides a more complete traverse of the probability-estimation tables.

Table 6 Images for parameter and marker variations.

Name	*Image count*	*Algorithm*	*Description*	*Purpose*
all0/all1	31	Both*	Uniform 0s data and uniform 1s data	Simplest input
AT	707	JBIG	Move AT to each tx [3:127]	AT unit
decode_only	5	JBIG	Truncated compressed data	ABORT marker code
superc	1	Both	Clustered dither halftone image	Example of halftone
dit	1	Both	Ordered dither halftone image	Example of halftone
erd	1	Both	Error diffusion halftone image	Example of halftone
panda	1	Both	Line-based halftone image	Example of halftone
pandaq	1	ABIC	Quantized line-based halftone image	Example of halftone
img4.bwimg	1	JBIG	G3/G4 focused test image	Vertical and horizontal patterns
height	1061	JBIG	Various vertical dimensions and stripes	Image vertical dimension
height	111	ABIC	Various vertical dimensions	Image vertical dimension

*"Both" indicates that the compressed data are available for JBIG and ABIC algorithms.

The image names that begin with the letters jamq are bit sequences that test the same algorithm features that the q patterns test; in addition, the context is generated by the model unit.

The patterns listed as stack, carry, fax21_carry, and fax31_stack are tests of the JBIG stack counter. The patterns called jbign and abic1 are special multiline patterns that test all possible template states in the middle of a line and on the left and right edges of the image.

The patterns called jbig_31 are examples of images for which the data expand on encoding (the compressed data byte count is greater than the source image data byte count). The patterns called trailing00 terminate with one or more bytes of 0x00. The trailing00 patterns are tests of the JBIG option that removes trailing 0x00 bytes from the compressed data during encoding and inserts any required trailing 0x00 bytes during decoding.

Table 7 JBIG vertical dimensions.

Image count	*Xd*	*Yd*	*Description*
528	1. . .8	1. . .33	Two-line and three-line
106	8	34. . .437	Two-line and three-line
413	1. . .8	1. . .437	Two-line and three-line with various lines per stripe
14	1, 9	3. . .6	ATMOVE

Table 8 ABIC vertical dimensions.

Image count	*Xd*	*Yd*	*Description*
50	8	1. . .50	Systematic change in vertical dimension
61	8	60. . .437	Random vertical dimension

• *Parameter and marker variations*

The images used for variation of parameters and markers are listed in **Table 6**. JBIG has an order of magnitude more patterns than ABIC to provide more thorough testing of the additional JBIG template and various AT pixel positions. (JBIG has a two-line template and a three-line template; ABIC has only a two-line template.)

The all-0 and all-1 bit sequences listed as all0/all1 are useful for testing the ability to simply process bits during encoding and decoding. The patterns listed as atmove systematically move the adaptive template (AT) pixel position along the current line to test the template implementation. The patterns listed as decode_only have the ABORT marker inserted into compressed data to confirm that the decoder will abort properly.

To test a variety of halftone types, a group of halftoned images are included from [14]. The halftone images are listed as superc, dit, erd, panda, and pandaq. The imag4.bwimg image was used to test the G3/G4 facsimile algorithms [12, 13]. The patterns listed as "height" provide bit sequences of various image vertical dimensions for both ABIC and JBIG.

Table 7 shows the distribution of the Xd and Yd parameters used for the JBIG "height" images listed in Table 6. For vertical dimension tests, bit sequences of convenient size were selected from the Q-value table/adapter unit, stack counter, and AT unit sets (shown in Table 5) and encoded with different parameter settings.

Systematic horizontal variation from one to eight bits and vertical variation from one to 33 lines provided 528 patterns. At the eight-bit horizontal dimension, 106 images were generated randomly with vertical dimension in the range 34 to 437 for both two-line and three-line templates with a single stripe per image (vertical dimensions tend to cluster around integer multiples of 32 for more complete testing of the implementation specifics of the IBM JBIG–ABIC core). Additional variation in image dimension, bit sequence, and coding options is available

Table 9 Randomly generated images.

Name	*Image count*	*Algorithm*	*Description*	*Purpose*
rnd_jbig	123	JBIG	Random image content, dimension, and options	Added test function coverage
rnd_abic	1120	ABIC	Random image content and dimensions	Added test function coverage

Table 10 JBIG randomly generated image dimensions.

Image count	*Xd*	*Yd*	*Description*
2	65,530	1	Extreme horizontal dimension
4	65,530	5	Extreme horizontal dimension
5	4096. . .65,530	5. . .12	Random data and dimensions
101	100. . .300	100. . .300	Random data and dimensions
5	5. . .12	4096. . .65,530	Random data and dimensions
4	5	65,530	Extreme vertical dimension
2	1	65,530	Extreme vertical dimension

Table 11 JBIG randomly generated image parameters and markers.

Option	*Probability or range*	*Description*
Stripes	{1. . .10}	Number of stripes varied randomly between 1 and 10
LRLTWO	50%	Two-line (50%) vs. three-line (50%)
ATMOVE	75%	Adaptive pixel moved (75%) vs. default position (25%)
TPBON	25%	Typical prediction on (25%) vs. off (75%)
SDNORM/SDRST	50%	Stripe end save statistics (50%) vs. reset statistics (50%)

with the 413 images having horizontal dimension in the range 1–8 and vertical dimension in the range 1–437. Multiple stripes are encoded in four of the patterns. The ATMOVE marker is used for 14 images.

Table 8 shows the distribution of the Xd and Yd parameters used for the ABIC height images listed in Table 6. The ABIC height images are a subset of those for JBIG. The horizontal dimension is fixed at eight bits. The vertical dimension is varied systematically from one to 50 and selectively from 60 to 437 (again clustered near integer multiples of 32 for more complete testing of the core implementation).

- *Randomly generated images*

The randomly generated images are listed in **Table 9**. The randomly generated images were selected such that the simulation time is acceptable with present-day systems by keeping the total bit count per image relatively small. Random images for JBIG vary the horizontal dimension (Xd), vertical dimension (Yd), lines per stripe (L0), template selection (LRLTWO), ATMOVE horizontal displacement (tx), typical prediction enable (TPBON), and the stripe termination marker (SDNORM/SDRST). The data content of the source images is random. **Table 10** shows the distribution of Xd and Yd for the JBIG random images.

Table 11 shows the marker and parameter distribution statistics of the images in Table 10. The images are composed of horizontal stripes. The number of stripes per image is uniformly distributed between one and ten stripes. The two-line template is used in 50% of the images, an AT move is used in 75%, typical prediction is enabled in 25%, and SDNORM stripe termination marker is used in 50% of the images.

Random images for ABIC have two parameters, the horizontal (Xd) and the vertical (Yd) dimensions. The horizontal dimension is specified precisely to the single pixel (bit), and internally the ABIC compression process pads the source image at the right margin with background (0) pixels such that the processed horizontal dimension is an integer multiple of eight pixels. **Table 12** shows the distribution of the Xd and Yd image dimension parameters used for the ABIC random images. The data content of the source images is random.

- *Implementation-specific images*

Specific design implementation particulars were tested in the IBM JBIG–ABIC core by encoding and decoding

Table 12 ABIC randomly generated image dimensions.

Image count	*Xd*	*Yd*	*Description*
1000	1. . .1030	4	Random data and horizontal dimension
20	2200. . .2400	1. . .200	Random data and dimensions
10	1500. . .1600	200. . .300	Random data and dimensions
70	2300. . .2400	200. . .300	Random data and dimensions
5	33,000. . .34,000	1. . .100	Random data and dimensions
5	1. . .200	4800. . .6000	Random data and dimensions
3	524,240	1	Large horizontal dimension
2	524,240	3	Large horizontal dimension
3	8	65,530	Extreme vertical dimension
2	16	65,530	Extreme vertical dimension

Table 13 Implementation-specific images.

Name	*Image count*	*Algorithm*	*Description*	*Purpose*
counter	24	JBIG	Random data and Xd = 255, 256, 257	DMA state machine
DMA	400	JBIG	Random data and Xd = n8	DMA unit
counter	12	ABIC	Random data and Xd = 255, 256, 257	DMA state machine
DMA	594	ABIC	Random data and Xd = n8	DMA unit

implementation-specific images. The implementation-specific images are listed in **Table 13**.

Some of the specific states of the state machines in the IBM JBIG–ABIC core were tested with 24 JBIG images and 12 ABIC images listed as "counter." To test the DMA unit of the IBM JBIG–ABIC core, 400 JBIG images and 594 ABIC images were designed. In the core, the DMA unit has four buffers of 32 bytes each that hold the compressed data, current line, previous line, and second previous line (JBIG three-line template). The DMA patterns may be of less use when testing implementations having different particulars.

Acknowledgments

Many individuals developed software and source image data that contributed to the IBM JBIG–ABIC Verification Suite. A key contributor was Michael J. Slattery of IBM Burlington, who wrote the original Qx "C" code implementation that was used to generate many of the coder-only tests and to debug IBM JBIG–ABIC core design errors. Another key contributor was Frank A. Kampf of IBM Burlington, who extended and enhanced the original Qx "C" code, produced most of the random images, and created useful images such as the image that requires QM-coder latency exceeding 32 KB. Ronald B. Arps of IBM Almaden provided the CCR image-compression software that was used to generate the reference encoded data. Corneliu M. Constantinescu, also of IBM Almaden, provided enhancements to the CCR software. The authors would also like to thank their management, Ted R. Lattrell and Frederick C. Mintzer, for support and encouragement for the JBIG–ABIC verification work. JLM especially appreciated the part-time employment at IBM Burlington during her leave of absence.

References

1. *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).
2. 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).
3. U.S. JBIG Committee (X3L3.2) peer-reviewed JBIG validation images.
4. K. M. Marks, "A JBIG–ABIC Compression Engine for Digital Document Processing," *IBM J. Res. Develop.* **42,** 753–758 (1998, this issue).
5. M. J. Slattery and J. L. Mitchell, "The Qx-Coder," *IBM J. Res. Develop.* **42,** 767–784 (1998, this issue).
6. *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).
7. W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1993 (ISBN 0-442-01272-1).
8. J. L. Mitchell and W. B. Pennebaker, "Software Implementations of the Q-Coder," *IBM J. Res. Develop.* **32,** 753–774 (1988).
9. 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).
10. *Data Stream and Object Architectures: Image Object*

Content Architecture Reference Manual, Order No. SC31-6805-01; available through IBM branch offices.
11. *ITU-T Rec. T.85, Application Profile for Recommendation T.82*—Progressive Bi-Level Image Compression for Facsimile Apparatus (JBIG fax application standard).
12. R. Hunter and A. H. Robinson, "International Digital Facsimile Coding Standards," *Proc. IEEE* **68,** No. 7, 854–867 (1980).
13. *ITU-T Rec. T.6 in Fascicle VII.3*, "Terminal Equipment and Protocols for Telematic Services," recommendations of the T Series, VIIIth Plenary Assembly, Malaga-Torremolinos, October 8–19, 1984.
14. Y.-H. Chen, F. C. Mintzer, and K. S. Pennington, "PANDA: Processing Algorithm for Noncoded Document Acquisition," *IBM J. Res. Develop.* **31,** 32–43 (1987).

Received February 16, 1998; accepted for publication October 2, 1998

Peter S. Colyer *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (pcolyer@us.ibm.com).* In 1982 Mr. Colyer graduated *magna cum laude* from Siena College, Loudonville, New York, with a B.S. degree in physics. He received a B.S. degree in electrical engineering from Clarkson College of Technology in 1983, and an M.S. degree in electrical engineering from Syracuse University in 1989. During the summer of 1982, Mr. Colyer worked at IBM in Poughkeepsie, New York, as a preprofessional engineer; the following year he joined IBM Poughkeepsie as a permanent employee. From 1983 to 1987, he designed and analyzed reliability tests for semiconductor memories. From 1987 through 1989, he redesigned and repackaged an *in situ* memory test and burn-in system. In 1990, Mr. Colyer joined the IBM BiCMOS ASIC development group in Burlington, Vermont. From 1991 to 1993 he led the debugging effort for an ASIC design system for the AS/400 CPU chip set. Since 1993, he has worked in the ASIC core macro development group. Mr. Colyer received an IBM Excellence Award in 1994 and a First Patent Application Achievement Award in 1995. He is a member of the Alpha Kappa Alpha, Sigma Pi Sigma, and Tau Beta Pi honor societies and is also a member of Mensa.

Joan L. Mitchell *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (joanm@us.ibm.com).* Dr. Mitchell graduated from Stanford University with a B.S. degree in physics in 1969. She received her M.S. and Ph.D. degrees in physics from the University of Illinois at Champaign–Urbana in 1971 and 1974, respectively, joining the Exploratory Printing Technologies group at the IBM Thomas J. Watson Research Center immediately after completing her Ph.D. She was a manager at the Research Center for nine years, worked for three years in IBM Marketing, and returned to the IBM Research Division in 1991 to work again in the Image Technologies group. In 1994, she left for a two-year leave of absence. During her leave, Dr. Mitchell co-authored a book on MPEG, consulted for IBM Burlington, and was a visiting professor at the University of Illinois for six months. Back at the IBM Thomas J. Watson Research Center, she is now a Research Staff Member in the Image Applications Department. Since 1976 Dr. Mitchell has worked in the field of image processing and data compression. She received IBM Outstanding Innovation Awards for two-dimensional data compression in 1978, for teleconferencing in 1982, for the image view facility in 1985, for resistive ribbon thermal transfer printing technology in 1985, for speed-optimized software implementations of image compression algorithms in 1991, and for the Q-coder in 1991. She is a member of APS, IEEE, Sigma Xi, and IS&T and is a co-inventor on 30 patents.