MorphoSys: a Reconfigurable Processor Targeted to High Performance Image Application

Guangming Lu [1], Ming-hau Lee [1], Hartej Singh [1], Nader Bagherzadeh [1],
Fadi J. Kurdahi [1], Eliseu M. Filho [2]

[1] Department of Electrical and Computer Engineering
University of California, Irvine, CA 92715 USA
{glu, mlee, hsigh, nader, kurdahi}@ece.uci.edu

[2] Department of Systems and Computer Engineering
Federal University of Rio de Janeiro   P.O.BOX 68511 21945-970 Rio de Janeiro, RJ, Brazil
eliseu@cos.ufrj.br

Abstract: This paper addresses the design idea of the MorphoSys Reconfigurable processor developed by the researchers in the UC, Irvine. With the demand to perform the multimedia operations efficiently, it is one of the directions that general processor needs to incorporate with some reconfigurable computing units, like FPGA. In MorphoSys project, we successfully propose a prototype to fulfill the above trend, which is comprised of a simplified general purpose MIPS-like RISC processor, called TinyRISC and 8x8 coarse grained reconfigurable cells, organized as SIMD architecture. MorphoSys is realized using 0.35um technology, and runs at 100Mhz with impressive performance enhancement compared with other architectures.

1. Introduction:

With the demand for the multimedia application, it is extremely necessary to explore another way beyond the general purpose processor to speedup the operation of the multimedia operations, such as compression, decompression, encryption, decryption, pattern recognition, target recognition.

In addition to the MMX technology, where new ISAs are added in the general processor to speed up the multimedia operation, recently, some processor architectures have been proposed to provide multiple configuration contexts in chip to program the LUTs and crossbars, like DPGA[1], or incorporate the general purpose processor with reconfigurable computing unit, such as GARP[2], MATRIX[3], RaPiD[4], RAW[5]. Basically, this kind of new architecture can be sub-divided into two categories:

1)   Fine-grained level reconfigurable unit, such as FPGA. Each bit is configurable.
2)   Coarse-grained level reconfigurable unit, such as reconfigurable array processor. Only the functionality and connectivity of each processor can be programmed.

In this paper, we describe a new reconfigurable processor named MorphoSys (Morphoing System), which is coarse level reconfigurable array processor. It is composed of a simplified version MIPS-like RISC, called TinyRISC, 8 by 8 Reconfigurable Cells (RC) Array, Frame Buffer, Context memory, DMA controller. We have developed VHDL version simulation environment and realized it by designing

custom blocks, such as 8x8 RC Array, Frame Buffer, context memory, Datapath in the TinyRISC and synthesizing other control logic.

This paper is organized as follows: Section 2 gives the brief overview about the MorphoSys system, and each major component. Section 3 introduces the M1 (MorphoSys Version 1) simulation environment MorphoSim, and the programmability of M1 chip. Section 4 gives the performance analysis of M1 architecture. Section 5 explains how we realize the M1 in physical design.

2. MorphoSys System

Figure 1 shows the components of the M1. It is composed of TinyRISC, 8x8 RC Array, context memory, frame buffer, DMA controller, instruction cache, data cache and memory controller. The following section will briefly explain the function and architecture of each component.
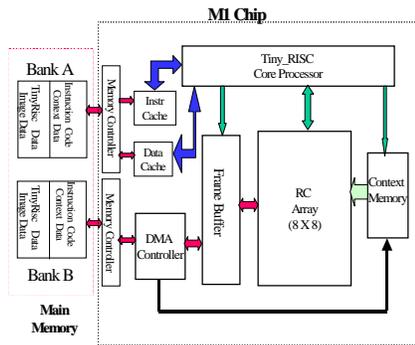

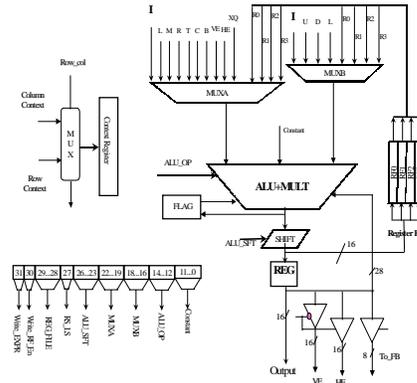Figure 1. System level architecture for M1 chip

Figure 2. Reconfigurable Cell Architecture

2.1 RC Array:
2.1.1 Architecture of Reconfigurable Cell

The reconfigurable cell (Figure 2) is the programmable unit of MorphoSys. It is coarse grained. The minimum programmable bitwidth is 16 bits. The RC Array (Figure 3) is consisted of 8x8 of identical RC. As showed in Figure 2, each RC comprises of an ALU-multiplier, a shift unit, and two multiplexers for ALU inputs. It has an output register, and a small register file. A context word, stored in the context register, defines the functionality of the RC. It also provides select bits to input multiplexers. In addition, the context word can also specify an immediate value (constant). One's counter (one part of ALU) is used to implement special functions, which require processing of binary image data, such as automatic target recognition (ATR).

2.1.2 Broadcast Context

This 32-bit register contains the context word to configure each RC. The programmability of the RC functionality and interconnection network is derived from the context word.

This context word is broadcast to RC in each row or column to achieve the data parallel operation. However, the RCs in different row or column may have different contexts applied to them. By switching row context broadcast to column context broadcast or vice versa, we can avoid data movement needed frequently in some applications, such as DCT. Meanwhile, it is also possible to enable only one specific row or column for operation in the RC Array. This feature is primarily useful in loading data into the RC Array. Since the context can be used selectively, and because the data bus limitations allow loading of only one column at a time, the same set of context words can be used repeatedly to load data into all the eight columns of the RC. This feature also allows irregular operations in the RC Array, for e.g. zigzag re-arrangement of array elements.
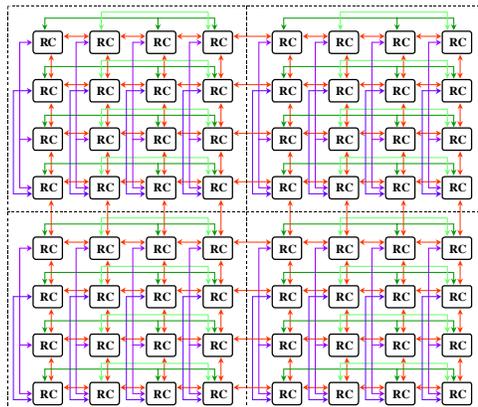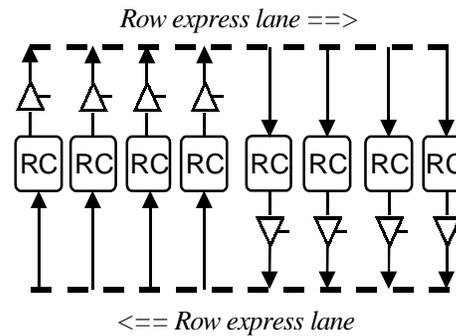

Figure 3. 8x8 RA Array

Row express lane ==>

<== Row express lane

Figure 4: Express lane connectivity (between cells in same row, but adjacent quadrants, same for the column express lanes)

2.1.3 Interconnection Network

The RC interconnection network is comprised of two hierarchical levels.

Intra-quadrant (complete row/column) connectivity: The first layer of connectivity is within one quadrant (a quadrant is a 4 by 4 RC partition). In the current MorphoSys specification, the RC array has four quadrants. Within each quadrant, each cell can access the output of any other cell in its row and column, as shown in Figure 3.

Inter-quadrant (express lane and nearby quadrant) connectivity: Between each pair of adjacent quadrant, the nearby quadrant connectivity exists in both vertical the horizontal direction. At the higher or global level, there are connections between adjacent quadrants. These buses also called express lanes, run across rows as well as columns. Figure 4 shows two express lanes going in each direction across a row. Therefore, any cell in any quadrant can access any RC output in the same row/column in the adjacent quadrant. The express lanes greatly enhance global connectivity. Even irregular communication patterns, that otherwise require extensive interconnections, can be handled quite efficiently. For example, an eight-point butterfly is accomplished

in only three cycles. The programmability of the connection is realized via the MUXA and MUXB in each RC, which is controlled by the configuration context. With these 2 levels connectivity, MorphoSys provides flexible interconnection to exchange data each other.

Data bus and Context bus: A 128-bit data bus from Frame Buffer to RC array is linked to column elements of the array. It provides two eight bit operands to each of the eight column cells. It is possible to load two operand data (Port A and Port B) in an entire column in one cycle. Eight cycles are required to load the entire RC array. The outputs of RC elements of each column are written back to frame buffer through Port A data bus. In order to perform row/column context broadcasting, there are 8-word-width context buses in both vertical and horizontal direction.

2.2 TinyRISC and MorphoSys Decoder

TinyRISC [6] is a simplified 32-bit MIPS RISC processor, which has four pipeline stages: fetch, decoder, execute, and write back. Since in this TinyRISC, it doesn't have the separate memory access pipeline stage, the memory address comes only directly from the register file. TinyRISC have 16 user accessible data registers and register number 0 is tied to zero. It uses a subset of conventional MIPS instruction set as well as some dedicated instructions used to control the RC Array, Context memory, frame buffer and DMA controller, etc.

MorphoSys Decoder, which is located at the decode stage, is dedicated to decode the MorphoSys instructions. Basically, It will either activate the DMAC to begin to transfer data between frame buffer, context memory with main memory, or let RC Array work through broadcasting configuration context. It establishes the communication between the general RISC processor, DMAC, Frame buffer, context memory and SIMD reconfigurable computing Units. Table 1 explains the some typical MorphoSys Instructions.

Table 1. Typical MorphoSys Instruction

LDCTXT	Load Context from Main Memory to Context memory.
LDFB	Load image data to frame buffer
STFB	Store the processed image data back to Main Memory
DBCBC	Column context broadcast, get image data from both banks
DBCBR	Row context broadcast, get image data from both banks
SBCB	Context broadcast, get one image data from a certain bank in frame buffer
CBCAST	Context broadcast, no data from frame buffer
WFBI	Write image data back to frame buffer with immediate frame buffer address
WFB	Write image back to the frame buffer, frame buffer address is from register
RCRISC	Write one data from RC Array to TinyRISC

2.2.3 Frame Buffer, Context Memory and DMAC

The frame buffer is consisted of 2 sets. Each set has 2 banks. Each bank has the 64x8 bytes storage. Bank A will provide the operand A for the RC Array, while Bank B will provide the operand B for the RC Array. The key feature of the Frame buffer is that once the starting address is given, it will activate 2 rows and read out the consecutive 8 bytes in order to provide 8 operands for the 8 RC's in a certain column. Meanwhile,

when one set is reading/writing data to/from RC, the other set can load/write data from/to main memory.

Context memory provides the SIMD-like instructions for 8x8 RC Array. All of the RCs in each column/row share the same context. Due to similarity of jobs each RC performs, we decide to centralize the context memory in each RC into column context and row context. They provide the context for column broadcasting and row broadcasting respectively. Obviously, the alternative solution is to make each RC have its context memory. So, each RC can perform different operation. But, it is the tradeoff between flexibility and chip size. Currently, we reserve 16 contexts for each column and row, since it will cover most of the contexts for MPEG2 application [7] mapping to M1 chip. Each context has 32 bits. The behavior and connectivity of RC is controlled by the context.

Basically, the DMAC handles all of the data movements involving RC Array with outside memory. It provides 3 atomic operations: from memory to Frame buffer, from memory to context memory and from frame buffer to memory. In fact, it is another state machine to handle the communication protocol between main memory and in-chip memory.

3. Simulation and Programmability of M1

MorphoSim is the VHDL version simulator for the MorphoSys reconfigurable computing processor. Using this simulator, it becomes feasible to verify the mapping algorithm, and validate the physical design. We have developed a compiler based on the SUIF compiler [8]. So, we can program M1 chip either on the C level (with in-line code) or on the assembly language level.
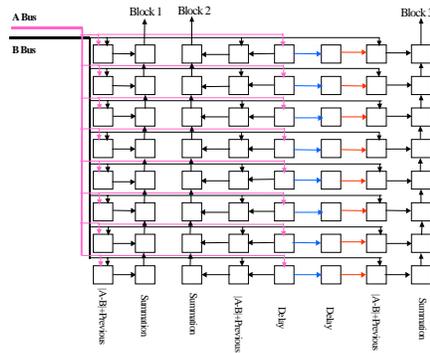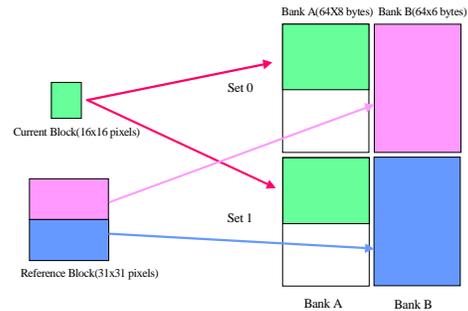

Figure 5. The Motion Estimation Mapping

Figure 6. Data Storage in Frame buffer for Motion Estimation

The following is one segment of C program example of Motion Estimation, which is compiled by the MorphoSys compiler, and generate the executable code for the M1 reconfigurable processor. In this mapping case, the RC Array can handle 3 blocks each time, generate the total sum of difference for each pixel, and send these 3 results back to the TinyRISC. TinyRISC will compare these 3 values with the previous minimum value, get the smallest one, then calculate the Motion Vector, finally, store it in the

register file. This Motion Vector will be used again in the Motion compensation in the MPEG2 application [9]. The data storage in the Frame buffer is visualized in the Figure 6. All the instruction with prefix TR_ will be executed in the RC Array. The rest of the instruction will be executed in the TinyRISC. Current version can't support automatic job partition between TinyRISC and RC Array. It should have the interference from users. But, this is an important sub-project of our current researches.

```
ME() {
  int H, V, minvalue, MVX, MVY, d1, d2, d3,  reg,  c1,bankstep, base;
# d1 get the data from Col#1; d2 get the data from Col#2; get the data from Col#2;
# set the minvalue as the biggest postitive number;
  bankstep = 4;   base = 0;  reg = 0; minvalue = 0x7FFFFFFF;
  for (H=0; H<16; H++) {
    for ( V=0; V<16; V=V+3) {
#####   The following instructions will be performed in RC Array
#              reg,  set, all, base,  ctx, Bank_B
        TR_dbcbc( reg,  0, 1,    0,     0,   0*bankstep);
        TR_dbcbc( reg,  0, 1,    1,     1,   1*bankstep);

        . . .

        TR_dbcbc( reg,  0, 1,    15,    2,   15*bankstep );
        TR_dbcbc( reg,  0, 1,    0,     3,   0);
        TR_dbcbc( reg,  0, 1,    1,     4,   0);
### Get the data generated in the previous iteration.
         TR_rcrisc (d1, 1);
         TR_rcrisc (d2, 2);
         TR_rcrisc (d3, 7);
#### choose the smallest among d1, d2,d3 and minvalue, then calculate the motion
Vectors, done in TinyRISC
        minvalue = min(d1, d2, d3, minvalue);
##      update the corresponding motion vector

      . . .
    } }
}
```

4. Performance Analysis Examples
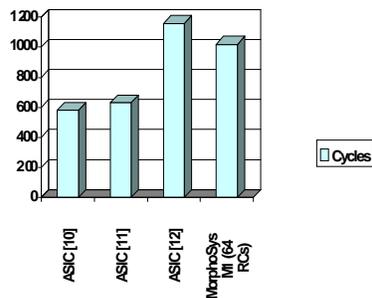
4.1 Motion Estimation

Figure 7: Performance Comparison for Motion Estimation
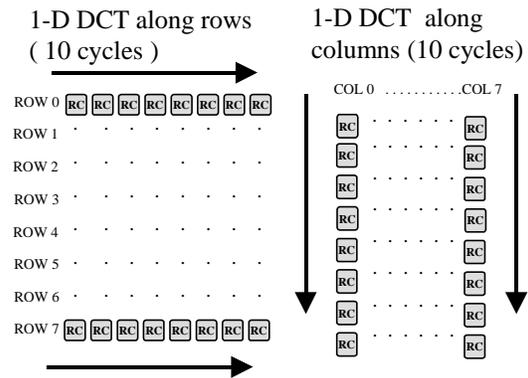

Figure 8: Computation of 2-D DCT across rows/columns (without data transposing)

The following comparison is based on 8x8 current block moving around on a reference block (search area) with 8-pixel displacement. Currently, full search algorithm is employed in the MorphoSys Motion Estimation mapping [17]. It is compared with three ASIC architectures implemented in [10], [11], [12]. The ASIC architectures have same processing units with MorphoSys. The figure 7 shows that the MorphoSys is comparable to the cycles required by the ASIC designs. Pentium MMX takes almost 29000 cycles for the same task, which is almost thirty times more than MorphoSys.

4.2 Two Dimension DCT

The Figure 8 shows the mapping of 8x8 pixels block to 8x8 RC Array. We broadcast the Horizontal context and Vertical contexts to perform Vertical DCT and Horizontal DCT. MorphoSys requires 21 cycles to complete 2-D DCT (or IDCT) on 8x8 block of pixel data [17]. This is in contrast to 240 cycles required by Pentium MMX ^{TM} [13]. REMARC [14] takes 54 cycles to implement the IDCT, even though it uses 64 nano-processors. The relative performance figures for MorphoSys and other implementations are given in Figure 9.
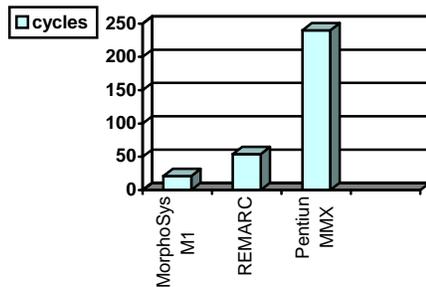
4.3 ATR Application

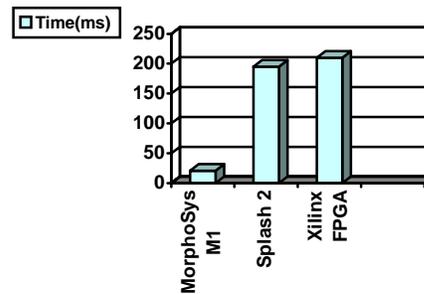Figure 9: DCT/IDCT Performance Comparison (cycles)

Figure 10: ATR Performance Comparison (ms)

Automatic Target Recognition (ATR) is to automatically detect, classify, recognize and identify an object. Under the same condition, for 16 pairs of target templates, Xilinx XC4010 FPGA [15] needs 210 ms processing time, Splash 2 [16] needs 195 ms, MorphoSys needs 21 ms (@ 100 Mhz) (Figure 10) [17].

5. Layout Realization of M1

We have finished the physical design cache and Register file in the TinyRISC, Frame buffer, individual RC using Magic. For the other pure logic circuits, we will use Synopsys to synthesize them, and use the AutoCells (from Mentor) to place standard cells and route. Since the rich connectivity and symmetrical property in 8x8 RC array, it is hard for commercial Router to handle the routing for 8x8 Array properly. Automatic

router will completely destroy the regularity, and make the clock H tree unbalanced. Therefore, we developed L language script file to finish the regular routing, and build the balanced clock tree (Figure 11). Inside RC, we only use Metal 1 and Metal 2 for internal routing, and reserve the Metal 3 and Metal 4 for connectivity among RC. Mentor Graphics MicroPlan and MicroRoute will handle the final routing on the top level, as showed in the Figure 1, including standard-cell blocks, such as DMAC, and custom design blocks, such as cache, RC Array. The final floor plan will be illustrated in the figure 12. We use the delay information gotten from MicroPlan to do back-annotation and use Lsim to do post-layout simulation.

6. Conclusions and future work

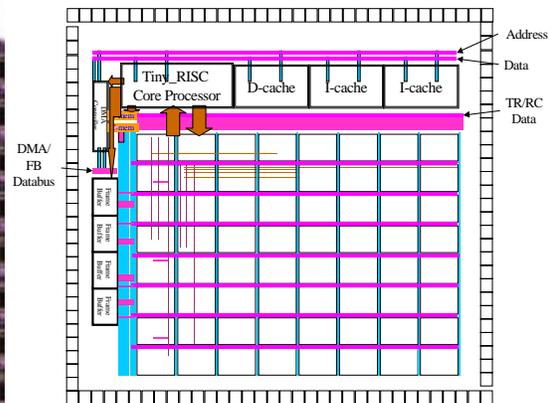Figure 11. Regular routing of 8x8 RC Array (12mmx10mm)

Figure 12. Final floor plan for the M1 chip (core 14mmx14mm).

In this paper, We have depicted the architecture and functionality of each main component of the MorphoSys and evaluated the performance of applications, such as Motion Estimation, DCT, ATR. Meanwhile, we have presented the simulation environment -- Morphosim and MorphoSys compiler, and will finally realize it in physical layout. We present a feasible way to integrate general-purpose microprocessor with an Array of reconfigurable cells. We can freely increase the size of the reconfigurable array based on the availability of the die area of the chip and performance requirement. MorphoSys is designed to be an independent system. But for M1, it has to rely on a host to communication with other peripheral devices. In current prototype, we download image data, executable code, context data through standard PCI bus, and upload the processed image data back to host to visualize it. The PCB design for M1 test chip is under development. Meanwhile, we will continue to enhance the current compiler to support the automatic job partition between TinyRISC and 8x8 RC Array.

7. Acknowledgments

References:

1) E. Tau, D. Chen, I. Eslick, J. Brown, A. DeHon, "A First Generation DPGA Implementation". *FPD'95 Third Canadian Workshop of Field-Programmable Devices, May 1995.*

2) J. R. Hauser and J. Wawrzynek, "Garp: A MIPS Processor with a Reconfigurable Co-processor," *Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines, 1997.*

3) E. Mirsky and A. DeHon, "MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources," *Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines*, 1996, pp.157-66.

4) C. Ebeling, D. Cronquist, and P. Franklin, "Configurable Computing: The Catalyst for High-Performance Architectures," *Proeedings of IEEE Internatinoal Conference on Application-specific Systems, Architectures and Processors*, July 1997, pp. 364-72.

5) J. Babb, M. Frank, V. Lee, E. Waingold, R. Barua, M. Taylor, J. Kim, S. Devabhaktuni, A. Agrawal, "The RAW Benchmark Suite: computation structures for general-purpose computing," *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, FCCM 97, 1997, pp. 134-43.

6) A. Abnous, C. Christensen, J. Gray, J. Lenell, A. Naylor and N. Bagherzaheh, "Design and implementation of the TinyRISC microprocessor" *Microprocessors and Microsystems. Vol.16, No.4, pp.187-94, 1992.*

7) Stanford PVRG-MPEG by anonymous ftp from havefun.stanford.edu:/pub/mpeg/MPEGv1.2.tar.Z.

8) SUIF Compiler system, The Stanford SUIF Compiler Group, *http://suif.stanford.edu*

9) MPEG2 Standard. http://www.mpeg.org.

10) C. Hsieh, T. Lin, "VLSI Architecture For Block Matching Motion Estimation Algorithm" *IEEE Transaction on CSVT, Vol. 2, June, 1992.*

11) S.H. Name, J.S. Baek, T.Y.Lee, M.K. Lee, "A VLSI Design For Full Search Block Matching Motion Estimation" *Proceedings of IEEE ASIC Conference, Rochester, NY, Set 1994.*

12) K-M Yang, M-T Sun, and L.Wu, "A Family of VLSI Designs for Motion Compensation Block Matching Algorithm" *IEEE Transaction on Circuits and Systems. Vol 36. No.10, Oct 89.*

13) Intel Application Notes for Pentium MMX. *http://developer.intel.com/drg/mmx/appnotes*.

14) T. Miyamori and K. Olukotun, "A Quantitative Analysis of Reconfigurable Coprocessors for Multimedia Applications*," Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines,* April 1998

15) J. Villasenor, B. Schoner, K. Chia, C. Zapata, H. J. Kim, C. Jones, S. Lansing, and B. Mangione-Smith "Configurable Computing Solutions for Automatic Target Recognition" *Proceedings of IEEE Workshop on FPGAs For Custom Computing Machines, April 1996.*

16) M. Rencher and B.L. Hutchings, "Automatic Target Recognition on SPLASH 2" Proceedings of IEEE Symposium on FPGAs for Custom Computing Machines, April 1997.

17) H. Singh, M. Lee, G. Lu, F. Kurdahi, N. Bagherzadeh, T. Lang, R. Heaton, E. Filho, "MorphoSys: An Integrated Re-configurable Architecture" *Proceeding of the NATO Symposium on Concepts and Integration, April, 1998.*