IEEE COMPUTER SOCIETY REPRINT

HIERARCHICAL CELLULAR LOGIC AND THE PIPE PROCESSOR: STRUCTURAL AND FUNCTIONAL CORRESPONDENCE

Ernest W. Kent
Steven L. Tanimoto

Reprinted from IEEE COMPUTER SOCIETY WORKSHOP ON
COMPUTER ARCHITECTURE FOR PATTERN ANALYSIS AND
IMAGE DATABASE MANAGEMENT — CAPAIDM
Miami Beach, Florida, November 18-20, 1985

IEEE COMPUTER SOCIETY
1730 Massachusetts Avenue, N.W.
Washington, D.C. 20036-1903

IEEE
COMPUTER
SOCIETY
PRESS

Hierarchical Cellular Logic and the PIPE[1] processor: Structural and Functional Correspondence.

Ernest W. Kent[2] and Steven L. Tanimoto[3]

National Bureau of Standards
and
University of Washington

Abstract

HCL is a hierarchical cellular logic, in which operations are applied to objects called bit-pyramids which themselves are functions on spaces called hierarchical domains. HCL provides an algebra for computations involving hierarchical, multiple-resolution representations of image data. PIPE is a newly-developed parallel architecture which supports multiple-resolution pyramid operations. This paper establishes that HCL is functionally equivalent to a subset of PIPE's instruction set, and that every HCL primitive operation corresponds to a single machine instruction in PIPE and executes in a single machine cycle. Further, the connectivity of HCL data-objects is embedded in the data-paths of the PIPE architecture. Thus, PIPE can operate upon the data-objects of HCL directly, without using extra storage for links or pointers, and without computation of storage addresses. As a result, PIPE programs implementing HCL may be expected to run enormously faster than corresponding programs for von Neumann machines, or for other parallel machines which do not share PIPE'S architectural correspondence to the structures of HCL.

Tanimoto (1984) has described a hierarchical cellular logic (HCL), in which operations are applied to objects called bit-pyramids which themselves are functions on spaces called hierarchical domains. HCL comprises an algebra for computations involving hierarchical, multiple-resolution representations of image data. As a formal system, HCL supports deductive analysis of image processing operations as well as the design of software for the operations. HCL provides expressive mechanisms for pattern-matching, resolution reduction, parallel processing and iterative processing of images. In addition to providing a formal theoretical basis for hierarchical cellular machines, HCL offers the possibility of efficient compilation of its expressions into programs for parallel machines with pyramidal architectures. Considered as a language, HCL compactly supports a large collection of useful image-processing and image-description capabilities which make explicit use of the hierarchical domain. We therefore are drawn to the task of implementing the operations of HCL on the most appropriate available hardware.

Ideally, such a machine should support the fundamental HCL operations as elementary machine operations, and should embed in its connectivity the connectivity of the HCL objects and domains. The first of these requirements follows immediately from considerations of operating efficiency; the ideal HCL machine should execute rather than simulate HCL operations. This implies the ability to specify hierarchical operations over pyramids and pyramidal neighborhoods in the micro-code of the device. The second condition follows from consideration of both operating efficiency and storage requirements. The representation of data objects in hierarchical domains typically requires multiple storage elements for every data item. In addition to representation of the data *per se*, a system of pointers is required to establish links to structurally related data items. The structure of the data object is implicitly contained in the connectivity established by this system of links. Since a single data item may belong to multiple data objects (e.g., pyramidal neighborhoods) in different relationships, it is generally not possible to represent this connectivity simply in an arrangement of data elements in address space. As a result, additional storage elements, or machine operations, or both, are required in order to represent or compute the connectivity of operations over multi-element data objects. Where these operations represent elementary HCL

1. "PIPE" is a registered trademark of Digital Analog Design Associates, Inc.

2. This work is the product of U.S. Government employees, and is not subject to U.S. copyright.

3. The contributions of S. Tanimoto were in part supported by N.S.F. grant MCS 8310410.

operators, this overhead can lead to considerable inefficiency. The ideal HCL device would employ data-flow paths which, by themselves, ordered the interactions of data elements and operators so that no storage elements other than data elements were required, and no algorithms were required to compute data element addresses.

The National Bureau of Standards' PIPE processor (Pipelined Image Processing Engine) supports a pyramid mode of operation. As such, it is to our knowledge the only full-scale pyramid machine in operation. The work reported here demonstrates that a subset of the pyramid-mode operations of PIPE are an exact embodiment of the fundamental logical operations of HCL. Furthermore, the data-flow paths within PIPE exactly support the connectivity of the HCL data objects, both within hierarchical domains, and between hierarchical domains which can exist contemporaneously in PIPE.

The PIPE image processor (Kent, Shneier, and Lumia, 1985) was inspired in part by pyramid concepts traceable to Tanimoto and Pavlidis (1975) and in part by designs previously developed in a collaboration between McCormick, Kent, and Dyer (1980, 1982). This latter work also influenced subsequent formulations of Dyer (1981), which had developed out of theoretical work with Rosenfeld (Rosenfeld 1979.) This work in turn contributed to Tanimoto's development of HCL. Thus it may not be surprising that there is an essential similarity between the fundamental operations of HCL, and the basic operations and organization of the PIPE processor.

The PIPE Architecture.

PIPE is composed of a series of modular processing stages (MPS). Each MPS contains both memory and processing capability. These stages are interconnected by three image-flow pathways (Figure 1). In general, the forward pathway carries processed images output from each stage (k) to the input of the subsequent stage (k-1). The retrograde pathway carries images output from each stage to the input of its predecessor (k+1), and the recursive pathway carries images output from each stage back into that stage's own input. At the input to each stage the information arriving on its three input paths may be combined according to any algebraic or Boolean operation to form a resultant input image for the stage. Within each stage, the image is stored in one of two buffers, from which it can be subjected to a variety of neighborhood and point operations. Each stage simuultaneously processes the image(s) contained in its buffers, completing the processing of an entire image (256 x 256 x

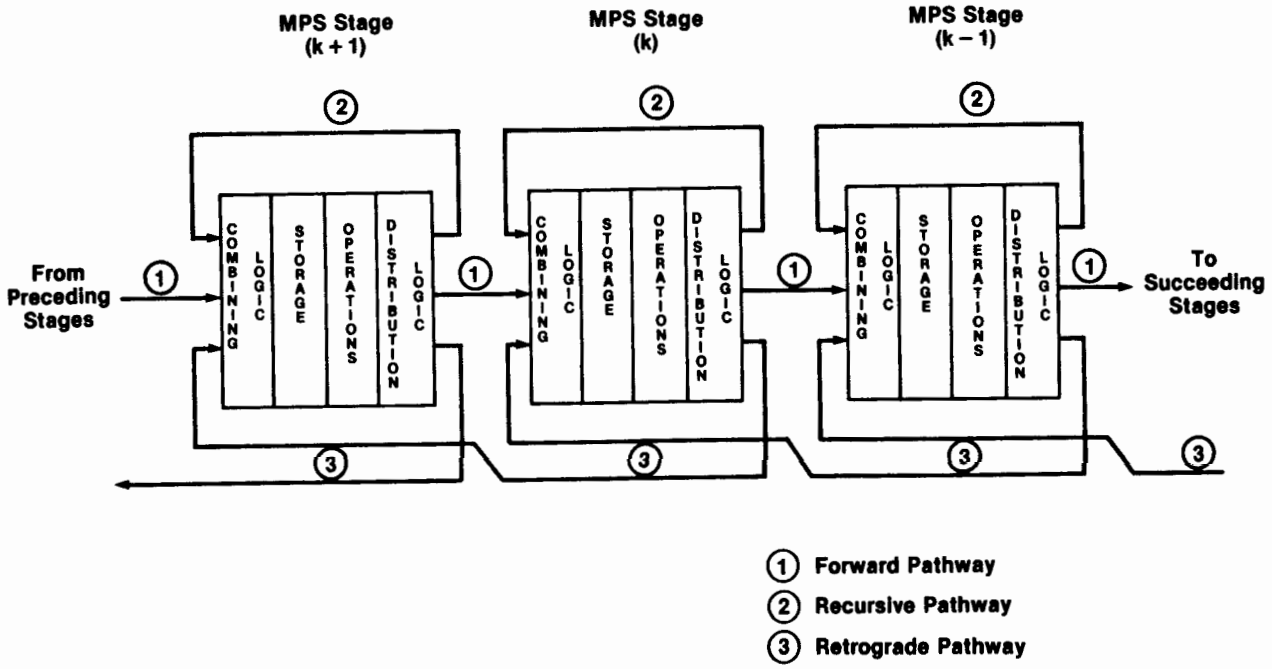

Figure 1. The PIPE processor consists of multiple modular processing stages connected by concurrent, interacting, image-flow pipelines. Results of independent operations on images in each stage are output over Forward, Recursive, and Retrograde pathways. The input to each stage may be obtained from any algebraic or Boolean combination of the images arriving on the three pathways.

8-bit) in one sixtieth of a second (one machine cycle.) The results of these operations, in any combination, can be output to the forward, retrograde, and recursive pathways leaving that stage. PIPE can perform a very large variety of interesting and useful operations on the images contained in the MPSs. Here we will concentrate only on a subset relevant to HCL.

When PIPE is operated in its pyramid mode, images passed over the forward pathway from stage (k) to stage (k-1) are reduced to one-half linear resolution (one fourth the number of pixels) by a sampling process which carries forward the odd-numbered pixels of odd-numbered rows into a densely-packed array. Images passed over the retrograde pathway are amplified by replication, so that each "father" in stage (k) is replicated into four adjacent "sons" in stage (k+1). Images passing over the recursive pathway are unchanged. In this mode, communications between bit-pyramids is handled in a consistent manner. The version of PIPE at the National Bureau of Standards contains eight MPSs, so that a complete hierarchical pyramid can be contained within the machine, starting from a 256 x 256 pixel image in the first stage, and tapering to a single pixel in the eighth. The PIPE stages (or any subset of them) may operate in non-pyramid mode to perform a variety of standard operations on images prior to their development into hierarchical representations. In particular, a variety of gray-scale operations may be performed on images prior to reduction to one or more binary images through thresholding on various derived properties.

The eight bits of a pixel may be treated within PIPE as an arithmetic quantity, in which case point and neighborhood operations produce appropriate arithmetic results. It is possible to deal with hierarchical pyramids of bytes in this fashion. However, it is also possible to treat the eight-bit pixel as eight independent bits of a Boolean vector. In this case, all point and neighborhood operations treat each bit-plane independently. Moreover, eight different and independent sets of point and neighborhood operations in a stage may be applied simultaneously to the eight bit-planes. With this capability, it is possible to develop and simultaneously operate upon eight independent hierarchical Boolean structures or, "bit-pyramids." Up to sixteen such structures can be handled if use is made of both of the image buffers within each stage. Interactions between different bit-plane structures may occur both before and after the within-bit-plane neighborhood operations, in the same machine cycle.

Pyramids, Hierarchical Domains, and Pyramidal Neighborhoods.

In HCL, Tanimoto describes a pyramid as a data object whose structure is a hierarchical domain which is in turn a set of cells. A cell is specified in HCL by the 3-tuple (k,i,j). A pyramid is defined as a function which maps each cell of a hierarchical domain to a value. A bit-pyramid is composed of elements whose values are in the range {0,1}, and a byte-pyramid is composed of cells whose values are in the range {0,255}. HCL defines a pyramidal neighborhood of a cell as consisting of eight same-resolution neighbors at level (k), one father cell at the next lower level of resolution (k-1), and four sons at the next higher level of resolution (k+1). This fourteen-cell neighborhood is illustrated in Figure 2.

HCL defines several operations on pyramids, which require operators to be applied to each cell of a pyramid, or to corresponding cells of multiple pyramids, generating a pyramid as a result. The result may occupy a new pyramid or replace one of its predecessors. In this class of operations, a value is generated for every cell in the input pyramid(s). By contrast, pyramidal neighborhood operations defined by HCL on the pyramidal neighborhood of a cell require that that cell's new value be computed by the result of an operation on all the cells of its pyramidal neighborhood. This result may replace the original contents of the cell or contribute to the construction of a new pyramid.

In PIPE, the cell values are stored in the bits (or bytes) of the pixels in the image buffers of the machine's MPS stages, and pyramids over the hierarchical domain are contained in a set of such buffers located in successive stages. Storage space for a bit-pyramid exists in the set of bit-planes with a common index over the set of buffers. Such a storage space will be referred to as a "domain register." The identity of an individual bit-pyramid (and the domain register containing it) will be understood from the name of the set of buffers (BUFF-A or BUFF-B) and the the bit-plane index. In a PIPE domain register, the index k then refers to the MPS stage and i,j to coordinates of the cell in the image buffer of that stage. In pyramid mode, the PIPE machine passes results of operations on the contents of every other cell of every other row of one of its stages forward into the next stage, where they are stored in a densely-packed array constituting the reduced resolution image of the contents of the preceding stage. This image constitutes the portion of the pyramid at the next level (k-1) of the domain. When this operation is performed over the machine's eight stages, a full 256

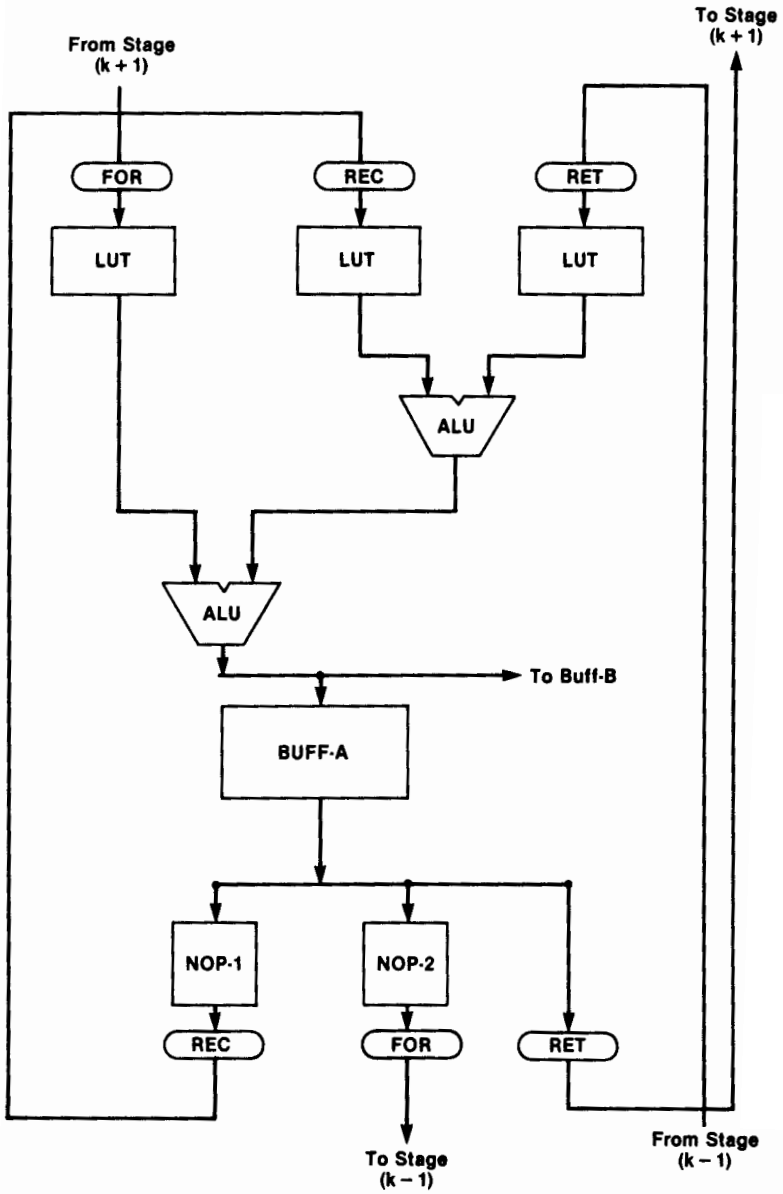

Figure 2. A pyramidal neighborhood of a cell in a hierarchical domain. The "home cell" of each such neighborhood has one father, eight lateral neighbors, and four sons.

x 256 array in the first stage is reduced to a single cell in the eighth stage, thus forming a complete pyramid.

AND_Match and OR_Match operations.

The two basic types of pyramidal neighborhood operations of HCL are called AND_Match and OR_Match. They require bit-by-bit comparisons between an arbitrary fourteen-cell pyramidal neighborhood pattern, P, consisting of 0, 1, or "don't care" (D) values, and the values of the corresponding cells of each pyramidal neighborhood of the hierarchical domain. This comparison has the value 1 under the AND_Match operation if all pattern elements (except the D's) equal their corresponding neighborhood values. The OR_Match operation returns 1 if at least one of the P-elements which is not a D matches its corresponding neighborhood value. The results of all individual comparisons in the neighborhood are then ANDed in AND_Match, and ORed in OR-Match, to generate the final value for the contents of the central cell of the pyramidal neighborhood. The same comparison pattern, P, is used in applying the operation to the pyramidal neighborhood of every cell of the hierarchical domain, so that an SIMD architecture suffices.

In PIPE, every 3 x 3 neighborhood bit-plane constitutes a nine-bit vector which addresses a result in a 512-entry look-up table. This table can contain the correct result of a match with P for all possible nine-bit neighborhoods, for either AND_Match or OR_Match operations. Every stage of the machine has the capability of performing two independent Boolean neighborhood operations of this sort simultaneously on the same data neighborhood. This may occur simultaneously and independently in every stage of the machine. In the same machine cycle, these results, or the data itself, may be communicated to other stages over the three inter-stage pathways described above. Using these resources, a cell and its lateral neighborhood may participate simultaneously in operations over three different pyramidal neighborhoods as a father, as a son, and as the principal element (home cell) of a pyramidal neighborhood. The method by which the PIPE architecture accomplishes this is illustrated by *reference to figure 3, which* schematizes several of the operating elements and data-paths of one MPS, a single PIPE stage. All of the data elements of a single hierarchical level are stored in buffers of such a stage (BUFF-A in the example of figure 3.) During a single machine cycle each of these elements and its lateral neighborhood will be sequentially accessed in raster-scan order,

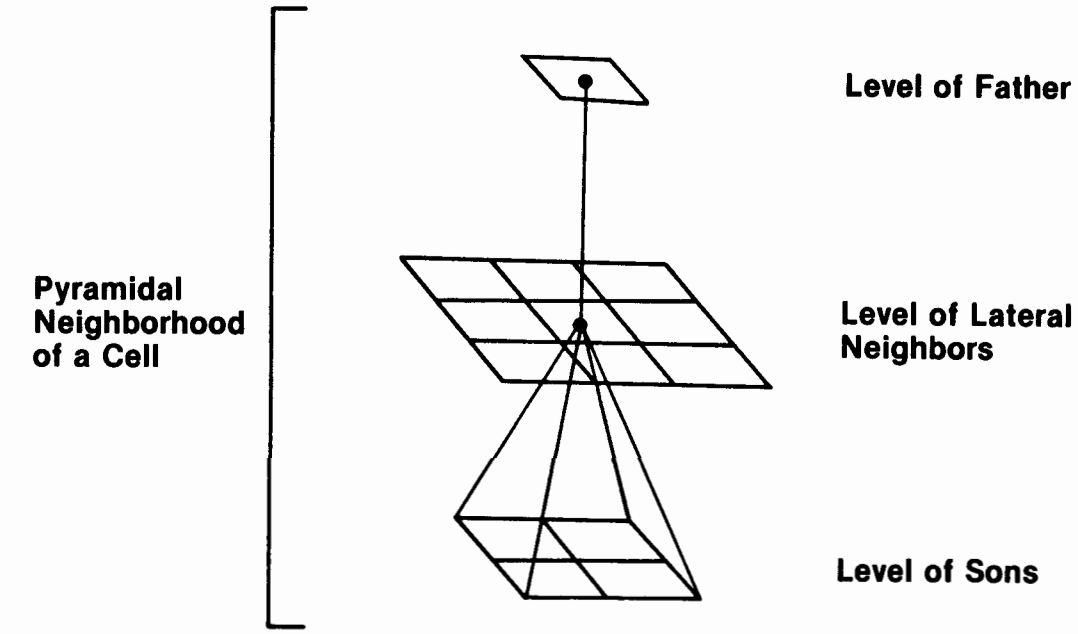

Figure 3. Schematic representation of some elements of a single modular processing stage (MSP) of the PIPE machine, showing operations of HCL. See text for description of elements.

and will participate simultaneously in three pyramidal neighborhood operations.

The result of the P-comparison operation over the home cell and its eight lateral neighbors is accomplished by routing the neighborhood data through the first neighborhood operator (NOP-1 in figure 3.) The result of the P-comparison for the same neighborhood considered as a neighborhood of sons is obtained by routing the neighborhood data through the second neighborhood operator (NOP-2 in figure 3.) In the case of son cells, only one of the four sons of any home cell is in direct upward communication with the home cell, due to the sampling procedure used to generate the half-resolution upward step. However, the neighborhood operation over the pyramidal sub-neighborhood occupied by the four sons may be used to generate the value passed up from the communicating son (without affecting the value stored in the communicating son.) This value will be the result of an AND or OR of the P-comparison operation over the sons' pyramidal sub-neighborhood. (At the level of the sons, the communicating son is the central cell of the 3 x 3 neighborhood, and only it and its east, south, and southeast neighbors may affect the result.) Because the AND and OR operations which combine the P-comparison operations over the pyramidal neighborhoods of HCL are commutative, the results obtained with this procedure do not differ from the result obtained if all the son cells communicate directly. The result of the P-comparison for the same neighborhood considered as the neighborhood of a father is simply obtained by routing the central element of the neighborhood directly out of the stage, since at the father level only the central cell of the neighborhood may affect the result. (The P-comparison for this single element is obtained after its receipt in the (k+1) stage.) In all of these operations, PIPE automatically supplies dummy neighborhood cells for incomplete neighborhoods at frame boundaries. PIPE has input and output stages at either end of the MPS chain which contain buffers which may be set to zero to supply a father of the apex and sons of the base.

As each sequence of results passes out of the neighborhood operators, these results are switched to the inter-level communication paths. The result of the lateral neighborhood-level P-comparison from NOP-1 is routed to the recursive pathway (labeled REC), and passes back into the input section of stage (k); the result of the son-level P-comparison from NOP-2 is routed to the forward pathway (labeled FOR), and passes to the input section of stage (k-1); the father data element is routed to the retrograde pathway (labeled RET), and passes to the input section of stage (k+1). The results arriving from different levels are combined by AND or OR operations at the input section of each stage. In figure 3, the input arriving on the RET pathway from stage (k-1) represents the father of a home cell in this stage (k). It is passed through a lookup table (LUT) which implements the P-comparison on the father element. The input arriving on the REC pathway is the stage (k) lateral-neighborhood result, which is passed unchanged through a LUT, and ANDed or ORed with the father result in an ALU. This result is further combined through a second ALU with the son result which arrives, after passing unchanged through a LUT, from stage (k+1) over the FOR pathway.

The final result is the value of the pyramidal neighborhood operation for a home cell in stage (k). Pipeline delays and read/write-operation address offsets are employed to secure the arrival of the result at such a time that it can, if desired, be written into BUFF-A of figure 3 without taking part in neighborhoods of cells still being read from BUFF-A. It is also possible to direct the result to a different buffer.

The pyramidal neighborhood operation described in the preceding paragraphs occurs simultaneously for all eight bit-planes, so that the eight bits of the home cell may contain the results of operations on up to eight independent bit-pyramids. Each of these may have been subjected to different pyramidal neighborhood operations, employing different P matches. At each stage of the operations, interactions among these bit-pyramids are possible by mapping the eight-bit element through LUTs, or passing it through ALUs. In particular, logical and barrel shifts, and other inter-domain register mappings may be used to accumulate results. For example, the result of an AND_Match or OR_Match operation on the domain register of bit 0 might be shifted left and result in a new domain register in bit 1, leaving the original domain register unaltered. Several 16-bit interactions are available so that the contents of both buffers may interact as sixteen independent domain registers.

Binary Operations.

A second class of operation in HCL requires logical interaction among bit pyramids of three hierarchical domain registers, designated X, Y, and Q. The binary operations over X, Y, and Q are pointwise logical operations among corresponding cells in different hierarchical domain

registers. The HCL operations specify a Boolean operation over X, or over X and Y, restricted by the state of domain register Q. All Boolean operations are permitted. If only X and Q are involved, any operation involving a Boolean function of one argument applied to X (such as NOT X) may be restricted to apply only where the corresponding cell of Q = 1. If X, Y, and Q are involved, a Boolean function of two arguments over X and Y (for example, X AND Y) may be similarly restricted to cells where Q=1..

The HCL binary operations may be carried out simultaneously at all levels of PIPE, on all elements of the domain registers, by passing the eight-bit contents of each cell through a look-up table containing all possible outcomes, and returning the result over the recursive pathway. As in the case of the pyramidal neighborhood operations, up to eight domain registers may interact, including the possibility of transferring the result to empty domain registers. Two such eight-domain register interactions may occur simultaneously at each level because of the presence of two complete buffers and operators. The results of the two eight-domain register operations may also be made to interact. In many cases, LUTs within the PIPE stages (omitted from figure 3 for clarity) will permit binary operations and pyramidal neighborhood operations to be combined in the same machine cycle.

Repetition and the "UNTIL NO CHANGE" Meta-operator.

The final set of fundamental HCL expressions governs the iteration of the operators already described. The essential requirement is to be able to repeat any operation, either for a fixed sequence, or until no further change occurs.

PIPE's stages are managed by a set of stage-control units which sequence the operations of the stages. The stage-control units can completely reconfigure a stage for a new operation between every machine cycle. The stage control units contain sequences of instructions, with repeat and branch points, for each stage. These sequences are down-loaded to the stage-control units from the host. Fixed repetition is thus accomplished. The "UNTIL NO CHANGE" meta-operator requires a simple comparison operation to be run between the result domain register and the previous result domain register after each iteration. This comparison may occupy a machine cycle for some HCL operations, but may be incorporated into the iteration for others. During the comparison operation, the stage control unit at each level can be made to detect the result of a failure to compare in any cell of any domain register, and to inhibit a sequence branch when this occurs during the machine cycle.

Extensions.

Some potentially interesting extensions to the capabilities of HCL can be envisioned using the PIPE hardware. The possibility of supplementing the hierarchical HCL operations with pre- or post-processing operations in non-pyramid mode has already been mentioned. Another potentially interesting extension would be the use of PIPE's arithmetic convolution mode of neighborhood operator to extend the scope of the functions defined in HCL over Boolean operators. Perhaps the most immediately useful extension involves PIPE's MIMD capabilities. Although HCL has been conceived as a formalism for SIMD machines, there is no inherent reason why different HCL operators could not be applied to different regions of an image in the same machine cycle on an MIMD machine. In PIPE, the contents of the bytes in either one of the two buffers in each stage can be interpreted as indices pointing to the set of tables and operations to be employed when processing the neighborhood of the corresponding byte in the other buffer. Thus, during each machine cycle, up to 255 different HCL operations could be employed over varying regions of the eight hierarchical domain registers contained in one of the buffers of each stage. (The selectable operations would be down-loaded by the host, along with the stage-control programs. The exact number of selectable operations available would depend on the amount of storage available for them, which is an option.) A typical use of this capability might be to vary the P-pattern of the pyramidal neighborhood operation over different image components such as edge and non-edge regions. Since the mapping is arbitrary, the indices in the directing buffer may in fact be data resulting from a previous image-processing cycle, such as an edge-detection operation.

Performance Analysis.

The potential performance of PIPE in executing HCL algorithms can be affected by several factors. Typically, execution time of HCL primitive operations will not be data dependent, since PIPE always operates over every neighborhood of every buffer in every machine cycle. Thus, the primitive operations are O(1), that is, a constant number of machine cycles. PIPE will perform more than 5×10^8 binary operations per second over cells of sixteen bit-domain registers. It can also perform 2.5×10^8 fourteen-neighbor pyramidal neighborhood operations per second over eight bit-domain registers. The actual efficiency obtained however depends on the way in which the data makes use of this potential.

The number of useful binary operations is diminished if all sixteen hierarchical bit-domain registers are not employed. Typically several will be employed in every operation, for example, as X, Y, and Q domain registers. Several such operations could be done in parallel, side by side in the sixteen available hierarchical bit-domain registers. Also, since PIPE's cycle time is designed to match RS-170 video field rates, progressive results of pyramidal binary operations on sequential images can be shifted laterally within the domain registers, and ultimately output at video rates after some sequence of operations while new images are taken in, thus maximizing use of the domain registers. If a straightforward approach to storage of pyramid constructions is employed, the number of useful 14-element pyramidal neighborhood operations is only 4.2×10^7 per second, due to the tapering of the pyramids. The effect is to replace a multiplication of the base area by eight levels with a multiplication of the base area by 4/3, which is the proportion of pyramid cells to image cells (Tanimoto and Pavlidis, 1975.) Storage schemes which could increase this number would appear to be difficult and marginally useful. As in the case of binary operations, the efficiency also depends on the extent to which multiple domain registers may be usefully subjected to pyramidal neighborhood operations simultaneously.

A factor which can be used to improve the average efficiency, is that there are many useful cases in which PIPE will be able to perform both binary operations and pyramidal neighborhood operations sequentially within each MPS, in the same machine cycle. This ability to micro-code multiple HCL primitive operations increases efficiency and, together with the ability to process multiple domain registers in parallel, can result in PIPE programs which have fewer machine cycles than HCL operations.

HCL programs can develop global image statistics useful in image processing. This requires bit-serial arithmetic operations developed from HCL primitives. Tanimoto (1984) demonstrates that the HCL computational model can implement bit-serial arithmetic in O(log N) time, provided that two hierarchical domain registers are available as "carry bits" in addition to one for the "sum." The HCL program to implement this arithmetic requires both pyramidal neighborhood operations which generate sums and inter-domain-register binary operations to deal properly with carry problems. In all, the operation inputs three bit pyramids, outputs three new ones, and employs two others in the computation process. This is an example of a useful operation in which the inter-domain-register binary operations may be scheduled within the same machine cycle as the pyramidal neighborhood operations. Thus, the PIPE program to implement bit-serial arithmetic over hierarchical domains may also be expected to run in O(log N) time.

Regardless of the actual processing speed of a machine, it will not be suitable for application to real-time image processing if images cannot be loaded and unloaded in real-time. This potential bottleneck must be considered in estimates of the machine's efficiency. In the case of PIPE, the basic operations are geared to television field rates (60 images/sec.) Multiple-resolution pyramids can be built at this rate as well, and, once such a hierarchy has been instantiated in the machine, it may be moved through the machine and replaced by subsequent pyramids at real-time rates if the image processing algorithms permit. Thus, HCL operations and not I/O considerations will be the true limiting factor in PIPE's pyramidal mode.

By way of comparison, the 4.2×10^7 fourteen-element pyramidal neighborhood operations that PIPE will perform each second could be expected to take 117 seconds on a von Neuman machine under the assumption that the machine could perform one access-operate-store instruction per microsecond, that another microsecond would be required for computation of indices, and that a factor of 10 could be saved by optimization techniques.

Conclusions.

Every fundamental HCL operation corresponds to at most a single PIPE machine instruction, and executes in a single machine cycle. PIPE can operate upon the data-objects of HCL directly, without using extra storage for links or pointers, and without computation of storage addresses. As a result, it should easily be possible to write a compiler for HCL which will permit all HCL statements to be translated into efficient PIPE programs. Such programs may be expected to run two orders of magnitude faster than corresponding programs for von Neumann machines, and also faster than other parallel machines which do not share PIPE'S architectural correspondence to the structures of HCL.

References

Dyer, C. R., A VLSI Pyramid Machine for Hierarchical Parallel Image Processing., Proc. PRIP '81, The IEEE Computer Soc. Conf. On pattern Recognition and Image Processing, Dallas, Tex., Aug. 1981, pp. 381-386.

Kent, E. W., Shneier, M. O., and Lumia, R., PIPE (Pipelined Image Processing Engine)., J. Parallel and Distributed Computing, 2, 50-78, 1985.

McCormick, B., Kent, E. W., and Dyer, C., A Visual Analyzer for Real-Time Interpretation of Time-Varying Imagery., in: "Multicomputers and Image Processing 3.", K. Preston and L. Uhr (Eds.), Academic Press, 1982.

McCormick, B., Kent, E. W., and Dyer, C., Highly-Parallel Structures for Real-Time Image Processing., Proposal submitted for Office of Naval Research SRO, 1980.

Rosenfeld, A., Picture Languages, Academic Press, New york, NY, 1979.

Tanimoto, S. L., A Hierarchical Cellular Logic for Pyramid Computers., J. Parallel and Distributed Computing, 1, 105-132, 1984.

Tanimoto, S. L., and Pavlidis, T., A Hierarchical Data Structure for Picture Processing, Computer Graphics and Image Processing, 4, 1975, 104-119.