ESD-TR-85-206

AD/

Semiannual Technical Summary

Restructurable VLSI Program

31 March 1985

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

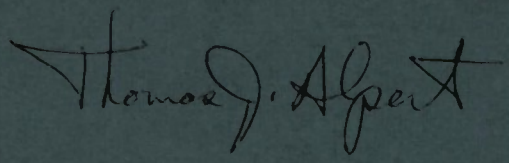
LEXINGTON, MASSACHUSETTS

Prepared for the Defense Advanced Research Projects Agency
under Electronic Systems Division Contract F19628-85-C-0002.

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract F19628-85-C-0002 (ARPA Order 3797).

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER



Thomas J. Alpert, Major, USAF
Chief, ESD Lincoln Laboratory Project Office

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

RESTRUCTURABLE VLSI PROGRAM

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

1 OCTOBER 1984 — 31 MARCH 1985

ISSUED 9 AUGUST 1985

LEXINGTON MASSACHUSETTS

ABSTRACT

This report describes work performed on the Restructurable VLSI Program sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency during the period 1 October 1984 through 31 March 1985.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

RESTRUCTURABLE VLSI PROGRAM

I. PROGRAM OVERVIEW AND SUMMARY

A. OVERVIEW

The main objective of the Lincoln Restructurable VLSI (RVLSI) Program is to develop design methodologies, architectures, design aids, and testing strategies for implementing wafer-scale systems with complexities approaching a million gates. In our approach, we envisage a modular style of architecture comprising an array of cells embedded in a regular interconnection. Ideally, the cells should consist of only a few basic types. The interconnection matrix is a fixed pattern of metal lines augmented by a complement of programmable switches or links. Conceptually, the links could be either volatile or nonvolatile. They could be of an electronic nature, such as a transistor switch, or could be permanently programmed through some mechanism such as a laser. The RVLSI Program is currently focusing on laser-formed interconnect.

The link concept offers the potential for a highly flexible, restructurable type of interconnect technology that could be exploited in a variety of ways. For example, logical cells or subsystems found to be faulty at wafer-probe time could be permanently excised from the rest of the wafer. The flexible interconnect could also be used to circumvent faulty logic and tie in redundant cells judiciously scattered around the wafer for this purpose. Also, the interconnect could be tailored to a specific application in order to minimize electrical degradations and performance penalties caused by unused wiring and links.

Further, the testing of a particular logical subsystem buried deep within a complex wafer-scale system poses a very difficult problem. A properly designed restructurable interconnect matrix could be temporarily configured to improve both the controllability and observability of internal cells from the wafer periphery. In this way, each component cell or a manageable cluster of cells could be tested in a straightforward manner using standard techniques. With an electronic linking mechanism, it is possible to think in terms of a dynamically reconfigurable system. Such a feature could be used to alter the function mode of a system to changes in the operating scenario, or it could be used to support some degree of fault tolerance if the system architecture was suitably designed.

Several major areas of research have been identified in the context of the RVLSI concept:

(1) System architectures and partitioning for whole-wafer implementation.

(2) Placement and routing strategies for optimal utilization of redundant resources and efficient interconnect.

(3) Assignment and linking algorithms to exploit redundancy and flexible interconnect.

(4) Methods for expediting cell design with emphasis on functional level descriptions, enhanced testability, and fault tolerance.

(5) Methods for testing complex, multiple-cell, whole-wafer systems.

Complementary work on the development of various link interconnect technologies as well as fabrication/processing technology is being supported by the Lincoln Air Force Line Program, and results are reported under the Lincoln Laboratory Advanced Electronic Technology Quarterly Technical Summary.

B. SUMMARY OF PROGRESS

Work for this period is reported under four headings: Design Aids for RVLSI (Section II), Applications (Section III), Testing (Section IV) and Link Technology (Section V).

1. Design Aids for RVLSI

Several improvements have been made in the wafer design and linking process. The design system has been enhanced to facilitate the restructuring of small segments of a wafer. The testing process has been improved to allow efficient use of test nets on a wafer when the testing and linking process is spread over several work sessions. The program for testing of the wafer interconnect has been improved by the inclusion of a more sophisticated model for track capacitance, and by a routine which automatically diagnoses the causes of track failures.

The first phase of the Digital Signal Processing Silicon Compiler is nearing completion. The compiler can now generate a complete resource specification and a delay-equalized netlist for simple, fixed-coefficient digital filters. The layouts of the multiplier, adder, and storage cells necessary for these designs are also complete. The remaining steps are to integrate these cell designs and the netlist with existing placement and routing routines to produce a completely automatic design system.

A general architecture for the second phase of the compiler has been developed. The second phase will allow the design of more sophisticated structures. In particular, it will allow sharing of computational resources to increase efficiency, and it will permit multiplexing and decimation of data streams. The details of the architecture are described.

2. Applications

The detailed design of the Dynamic Time Warping (DTW) wafer is underway. The simulations of the wafer have been completed and the necessary questions about the architecture and design parameters have been resolved. An overall layout for the entire wafer has

been completed. The detailed layout of one of the two cell types on the wafer is complete and has been submitted to MOSIS for fabrication. A support system, which will be able to demonstrate the wafer performance in various speech recognition tasks, is being designed.

3. Testing

The initial lot of optical probes has been distributed and a second lot is being fabricated.

4. Reduced-Dimension Laser Links

The results of some recent experiments on reduced-dimension laser links are reported.

II. DESIGN AIDS FOR RVLSI

A. LINKING SHELL (LSH)

Several enhancements have been made to LSH, the set of CAD programs used for design of each unique RVLSI wafer. The enhancements have been made to assist in incremental restructuring and testing and to improve analysis of interconnect.

In general, it is desirable to do assignment and linking for the entire wafer so that it is known that a restructuring is possible. In practice, however, an RVLSI wafer is restructured and tested incrementally so that if faulty components develop during the linking, they can easily be identified and replaced. To do this, the designer must be able to specify subsets of the system and the sequence in which they are to be restructured. LSH has permitted the generation of laser control files for a partial system by specification of a list of cells or a net file. But this is not flexible enough and the designer has had to generate special net files. The "zap" program has been changed to allow the user to specify a window on the physical wafer in which zap and cut commands will be generated.

The generation and use of test nets has been made more efficient. In the 31 March 1984 Semiannual, an LSH enhancement was described which generates test nets for testing of restructured interconnect and circuitry. It is desirable to link a large net in several pieces but, in order to conserve interconnect resources, do so with only one test net. The "ictest" program has been modified to support this operation. These improvements are sufficient for the integrator and FFT systems.[1] For testing the DTW wafer, the program will be further enhanced to permit incremental system testing of the wafer as it is linked.

A new interconnect analysis program, "analyz," has been installed replacing "analyze," the earlier version. This LSH function, which identifies good and bad tracks based on a statistical analysis of measured capacitance data, now has the ability to determine and explain why a track is bad. "Analyz" examines the location, the measured capacitance, and the predicted capacitance of each track in order to make its determination. The explanations for faults which "analyz" offers include: short to adjacent track, short to crossing track, and scratch on the wafer. Along with its explanation, "analyz" indicates how certain it is of its conclusion.

An experimental version of "analyz," which makes fewer assumptions about variation of capacitance across the wafer, has been written. The production version of "analyz" assumes, for the sake of simplicity, that track and link capacitances exhibit a linear variation with respect to coordinate location on the wafer. The experimental "analyz" makes no assumption about the nature of this variation. Instead, the program partitions the wafer into a mesh of rectangles and computes the capacitance parameters separately in each rectangle. Working with relatively simple wafer circuit layouts, the two versions of "analyz" produce essentially the same results. However, with wafer designs which exhibit denser and less regular geometric patterns of tracks and stubs, the more general point of view of the new "analyz" may prove essential for obtaining accurate results.

B. DSP SILICON COMPILER

The DSP silicon compiler is designed to convert signal processing algorithms, described in a high-level language, directly into a CIF file for circuit fabrication. The compiler can be used to generate either discrete chips or cells which can be incorporated into wafer designs. The designs consist of a collection of bit-serial arithmetic units plus the necessary storage and control circuits.

The current development effort has been divided into two phases. Phase I will produce circuits with fixed topology and fixed assignment of resources. It will produce fixed digital filter structures. Phase II will use the same types of resources but will generalize the ways in which they can be interconnected and controlled. This will permit sharing of the resources within the chip among several functions and will allow signals to be multiplexed at the input and output. The architecture for these chips will also allow the compiler to be extended to perform more general functions.

1. Phase I

The front end of the Phase I silicon compiler has been completed. It accepts as input a high-level description of an arbitrary, fixed-topology, fixed-coefficient filter and produces an output file containing a complete netlist of the filter describing both the data and control interconnect. Work will begin shortly on converting this file into a CIF file.

Delay Equalization

Data streams experience a computation delay when processed by serial arithmetic elements. This delay or latency depends on the type of arithmetic element (multiplier, adder) but is otherwise constant. Two data streams impinging on, say, an adder may have undergone different amounts of delay in arriving there and, therefore, must be brought into alignment before being processed. The process of achieving this alignment in a serial digital network is termed delay equalization.

During the past six months, an algorithm for automatically equalizing the delays in an arbitrary, fixed-topology digital network has been developed. The procedure begins by representing a digital network as a directed graph comprised of the elements given in Figure 1. The graph represents an adder graphically as a node followed by an arc having a delay equal to the **negative** of the adder's latency. An explicit delay element, on the other hand, is represented as a nodc followed by an arc having that delay. A simple network is depicted in Figure 2(a) along with its associated graph in Figure 2(b), drawn for $\ell_a = 1$ and $\ell_m = 24$. The graph corresponding to a network with negative delays to compensate for the latencies of the arithmetic elements is a logically correct, although physically unrealizable, representation of the original network. The process of delay equalization consists of manipulating the graph in ways that preserve its functionality but result in a new graph without negative delays.

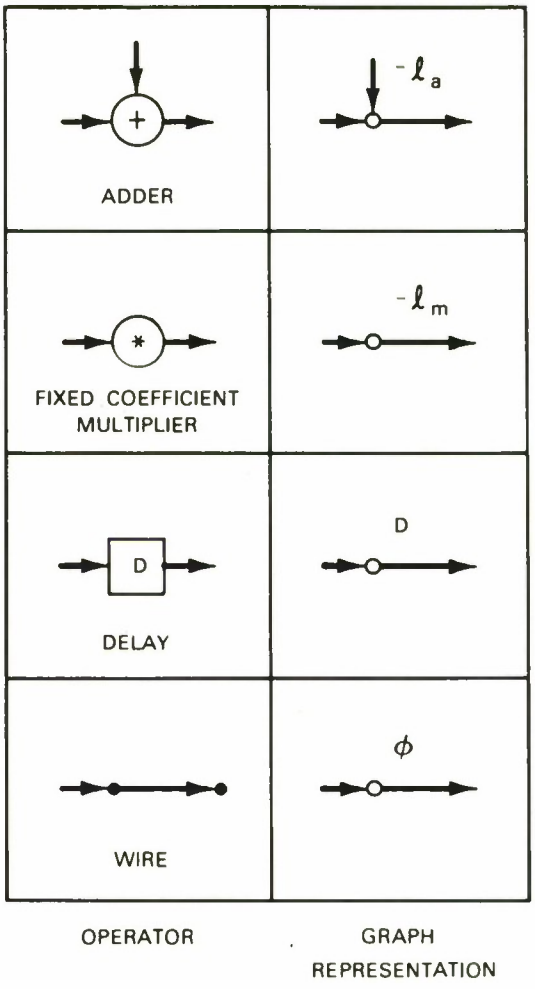


Figure 1. Graph representation of digital elements.

153347-N

A technique called retiming developed by Leiserson, Rose and Saxe[2] in a context apart from delay equalization can be employed to provide a solution to this problem. Specifically, the Bellman-Ford[3] algorithm employed by these authors in their retiming work can be used to solve the delay equalization problem. Applied to a given graph, this algorithm either concludes that delay equalization is impossible or else yields a specific assignment of delays to the arcs of the graph such that each arc has a non-negative total delay. The functionality of the graph is maintained by this delay assignment. The only case where equalization is impossible occurs when there is a feedback loop whose total delay is negative, indicating that the computational latency is so great that the fed-back result is not available in time for the next computation.

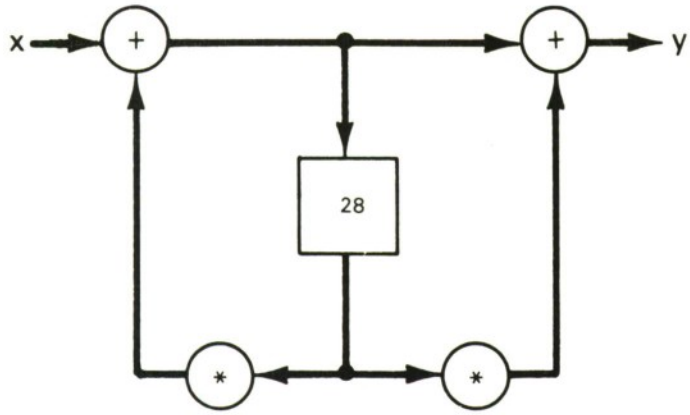


Figure 2(a). A simple network.

153348-N

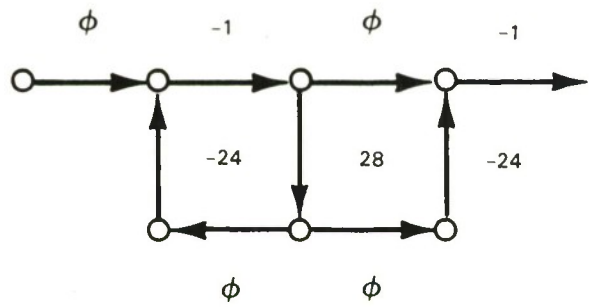


Figure 2(b). Network graph.

153349-N

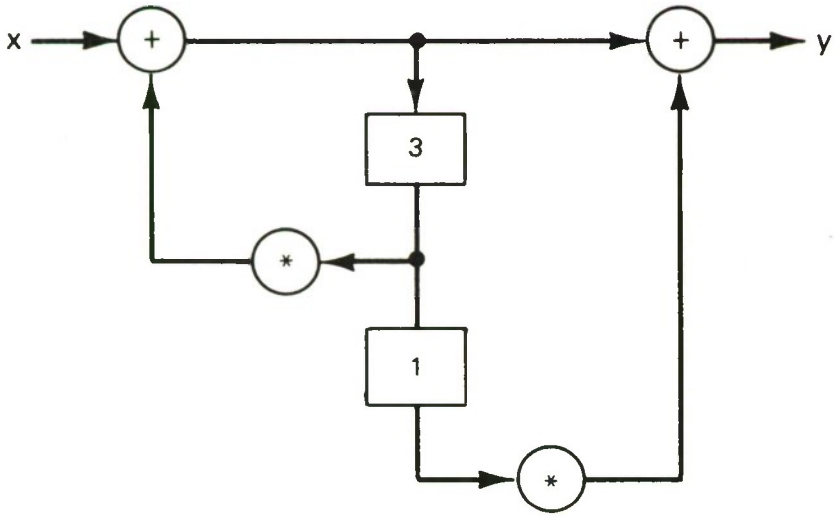


Figure 3. Delay equalized network.

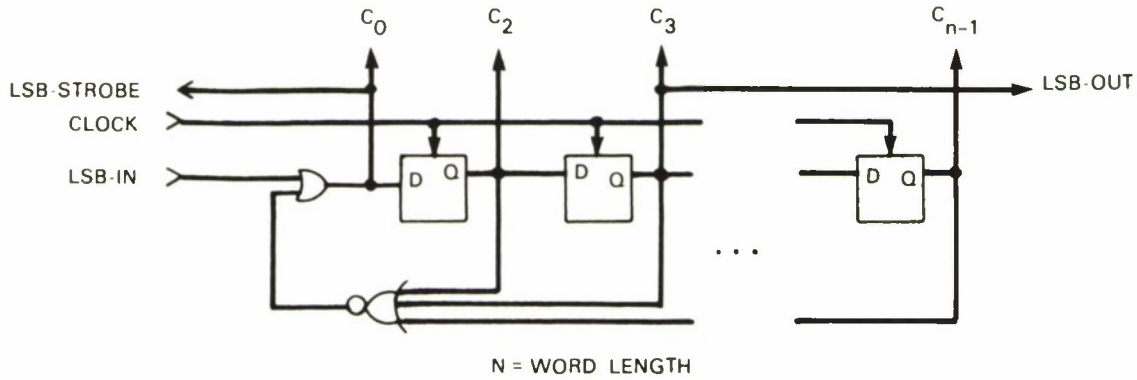
153350-N

This procedure has been coded and debugged and is now part of the compiler. The compiler produced the equalized network shown in Figure 3. The automatic equalization algorithm has been tried on a variety of examples and so far has always produced a delay equalization as good as any produced by hand.

Control Generation

The control unit of the compiled design must provide the necessary control signals to each functional unit and to the external circuits to which the chip is interfaced. In the case of the serial arithmetic units the required control is a one-bit signal coincident with the lsb of the word being processed. The external control on these chips will operate in either of two modes. The chip can be synchronized with the external circuitry by external frame pulses or the chip can generate the required pulse to synchronize the external circuitry. This flexibility also permits chips to be connected in tandem.

The logical design of the Phase I control unit has been completed. A diagram of the unit appears in Figure 4. The design can be described as self-initializing, synchronizable ring



153351-N

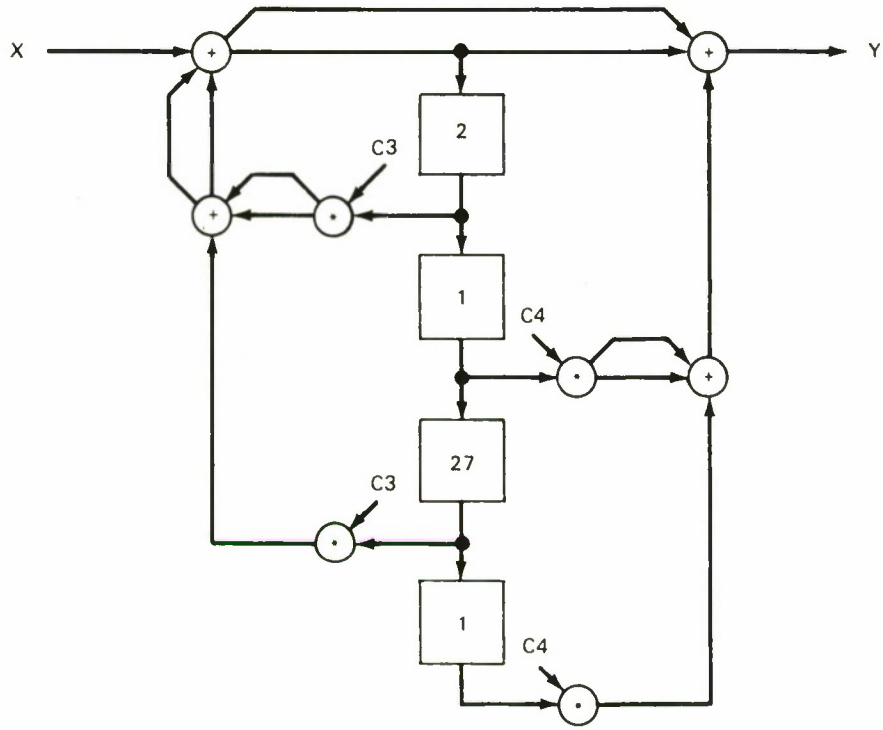
Figure 4. Phase I control unit.

counter. When lsb-in is held low, the feedback logic causes the counter to clear itself and then inject a "one" into the left-most flip-flop of the chain. This single "one" travels to the right-hand end of the chain and then gets clocked out of the system. At this point the feedback logic again injects a "one" into the chain and the cycle repeats. Operating in this mode, the controller circuit generates the desired lsb signal C_0 as well as all possible delayed versions of C_0 namely, C_1, C_2,...C_{n-1} which can be used to control adders, subtracters and multipliers.

The controller can also be synchronized to an external lsb signal by injecting this signal at lsb-in. It is easy to see that the counter will again clear itself and then fall into step with the external lsb-in signal. The controller also supplies two signals to the outside world, lsb-out and lsb-strobe. The signal lsb-out indicates when the lsb of the output data stream

is present and is shown in Figure 4 as C_2. The particular C_k that coincides with lsb-out depends on the filter being realized and is determined by the compiler. Lsb-strobe, shown in Figure 4 as C_0, is supplied to the outside as an indicator of when the chip expects the input data lsb to be present on its data line. Under user control, lsb-strobe can be made equal to any desired C_k.

The control signals generated by the controller are distributed to the arithmetic elements in a way that attempts to minimize the connections directly from the controller and maximize the use of control storage within the elements. This is shown in Figure 5, which



153352-N

Figure 5. An example of control distribution.

shows the control distribution for a second-order section. The signals C_3 and C_4 come directly from the controller. The control signal output from an arithmetic element is used to control a succeeding arithmetic element wherever possible. The algorithmically-derived control scheme shown in Figure 5 utilizes this possibility to the greatest extent possible.

The algorithm for assigning the control interconnect begins by traversing the delay equalized network looking for delay line outputs. The proper signal from the control generator is then connected to all arithmetic elements connected to this delay line output. The selection of the particular control signal to use in this step is derived from a delay potential function that was already computed while doing the delay equalization. The delay potential

is essentially the delay from a reference node in the network (usually an input pin) to all other nodes and thus contains the information needed to determine the proper control signal to send to a given node.

The assignment algorithm continues its work by recursively searching the output interconnect lists for each arithmetic element encountered in the last step for arithmetic elements whose control inputs are as yet unconnected. Such inputs on the new elements are connected to the arithmetic element control output of the previous step. The algorithm finishes up by making another pass through the network looking for uncontrolled arithmetic elements and connecting them to the proper control generator output. No claim of optimality in any sense is made for this algorithm, but in the cases tried thus far it seems to do an excellent job.

Cell Design and Layout Issues

The layouts for most of the cells needed for the Phase I compiler have been completed. These are the shift registers, adder, subtracter, and several multiplier cells. All these designs include the extra circuits required for feeding through the control signals and clocks. The circuit for the control unit has been designed, but the layout remains to be done.

The routines for the automatic generation of large functional blocks based on the cells are being tested. Two types of parametric units have been defined: arbitrary length shift registers and fixed coefficient multipliers. These programs are written in L5, a LISP-based language for integrated circuit layout developed at Lincoln Laboratory.[4] The parameter for the shift registers is the bit length and for the multiplier the binary expansion of the coefficient. The programs place all the interconnect between the cells, all the control signals, and all the clock and power lines.

Multiplier Units

An improvement has been made in the design of the serial multiplier. The bit-serial unit described in[5] assumes the two top bits in the data are equal, thus restricting the dynamic range of the data. If this condition is not met, the top bit in the result is not correct. In numbers represented in two's complement notation, the most significant bit carries a negative sign. This fact is the cause of the improper operation of the multiplier when the two last bits of the data are not equal. The unit was modified to solve this problem. By adding two pass-gates, we can alter the carry calculated by the unit's adder when the top bit is present (as this carry will not be needed) and produce the correct sign extension. This result is saved in a new register and now we are able to select the correct partial product being calculated by each unit the same way as before.

2. Phase II

Work on the Phase II compiler falls into two categories, architectural and algorithmic. All designs described by the signal processing language will be transformed into an architecture, i.e., hardware framework, which will be described. The following goals guided the development of the architecture:

- Low control-wire-density method of interconnecting functional units.
- Compact storage of state and temporary variables;
- Simple and versatile control units.

A block diagram of the architecture at the highest level is shown in Figure 6; each of the architectural components, i.e., functional units, storage, interconnection and control unit will be discussed individually below.

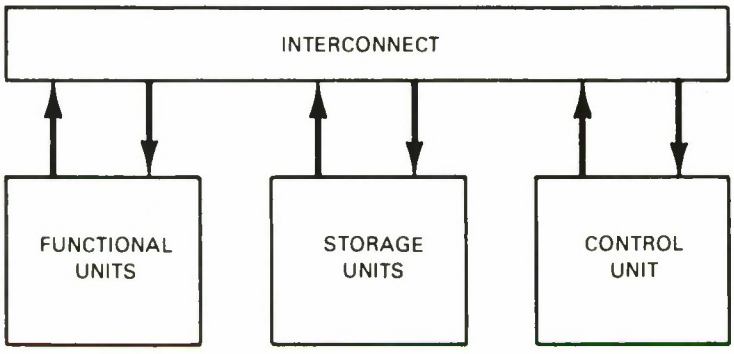


Figure 6. General architecture.

153353-N

Algorithms are being developed to transform the signal processing language into a hardware design which is a particular instantiation of the general architecture of Figure 6. The goals of algorithm development are to make the transformation in a way which minimizes the number of storage units and the complexity of interconnection. The algorithms themselves should be efficient, i.e., the programmed algorithm should run in time and space which is polynomial with respect to problem size. The discussion below will first concentrate on the architectural components shown in Figure 6, describing each in turn. Following that, the transformation algorithms will be discussed and finally, a summary of the work completed on the algorithms will be presented.

Functional Units

Functional units perform serial arithmetic and have serial-in and serial-out interfaces. Each functional unit also has a built-in auxiliary shift register which allows a control bit to shift through the functional unit in synchronism with the lsb. Two categories of functional units are defined, *inexpensive* and *expensive*. The expense of a functional unit is directly related to the silicon area necessary to realize that functional unit. Expensive functional

units, such as multipliers, must be used efficiently, i.e., a minimum number of them must be used and they must be kept as busy as possible performing useful calculations. The number of inexpensive functional units, such as adders, will be allowed to increase above the minimum specified by the signal processing in order to achieve a more efficient overall design.

Storage

Storage for state variables, temporary variables, and coefficients is provided by the two types of storage shown in Figure 7. The read/write storage unit is a set of registers physically contiguous in silicon. An operand to be written into a storage unit is shifted into a

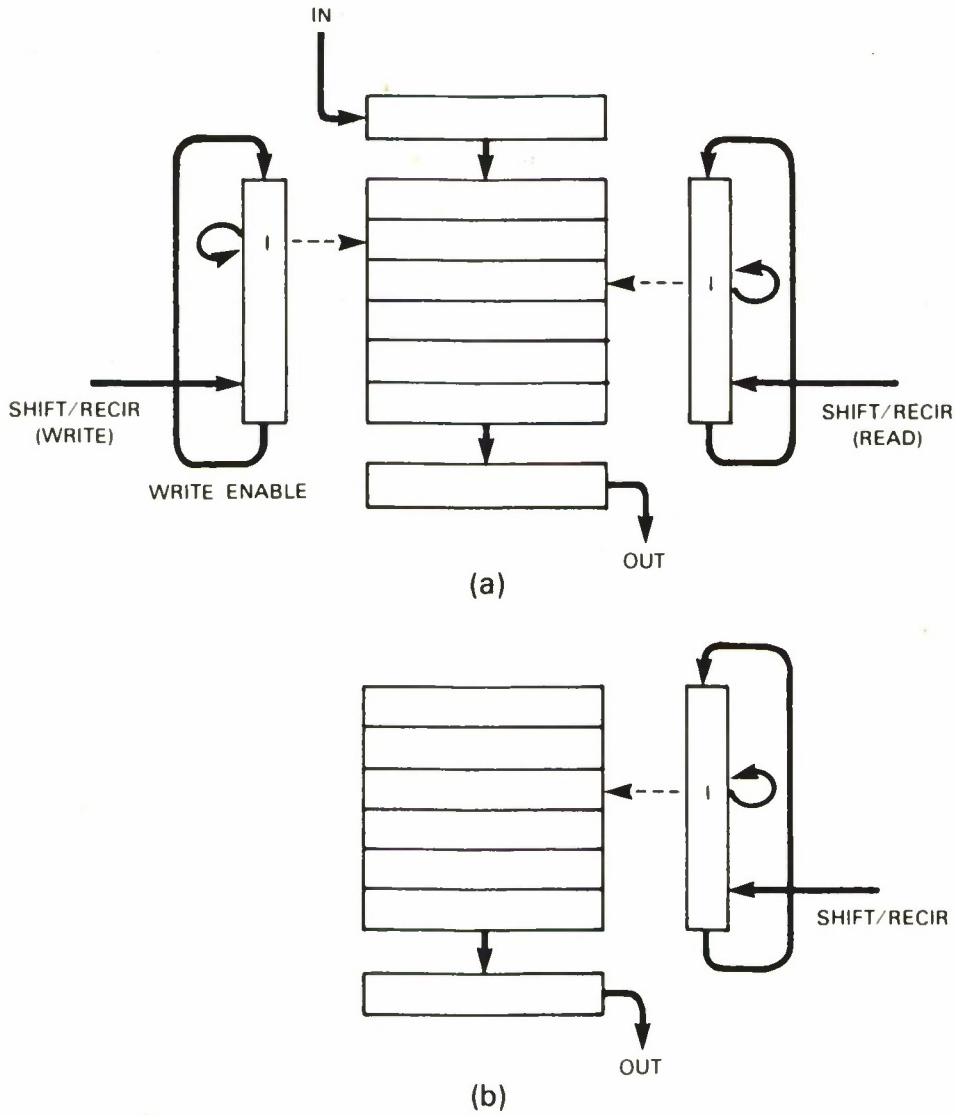


Figure 7. Storage units.

153354-N

serial shift register, and if the write is enabled, data words are written in parallel to one of the registers in the memory. The register to be written is selected by the 1 in a recirculating shift register shown on the left of the read/write storage unit in Figure 7; similarly, a register to be read is selected by a 1 in the shift register on the right. Data to be read is transferred in parallel to the output serial shift register and shifted out. Both the read and write recirculating shift registers work in a similar fashion. A signal from the control unit issued during each word time causes the 1 to remain stationary (SHIFT/RECIR signal = 0) or to proceed to the next shift register stage (SHIFT/RECIR signal = 1). The read-only storage unit is the same as the read/write storage unit except that hardware associated only with the write function is omitted.

This method of storage appears to satisfy storage needs for a wide range of signal processing applications and offers two advantages over a more conventional addressable RAM or ROM register file. The storage units presented here require no address decoding logic, resulting in a significant silicon area saving. In addition, control of individual storage units requires only 2 bits for a read/write storage unit and 1 bit for read-only storage unit, resulting in a shorter microword and fewer signal lines to be routed to the storage units. An additional global signal line is also needed by all read/write or read-only storage units to synchronize reads and writes with the input or output shift registers. The Write Enable signal for a storage unit is generated locally by ORing the states of its input connection switches. Connection switches will be described later. An algorithm to partition the storage requirements of a problem into units of this type has been developed.

Interconnection

Interconnection is a much more difficult problem in Phase II than it was in Phase I. When functional units and storage units are shared, a method must be found to dynamically switch connections when a shared functional unit changes uses.

In order to simplify the design we impose the following design rules:

(1) Connections are changed only at the word boundaries.

(2) The control settings for the interconnect elements are distributed serially during the word interval to all switches and can take effect on the action of one global timing signal.

The interconnection will be described by means of an example.

Figure 8 describes a second-order-section filter bank as an explicit loop in the signal processing language. The compiler will realize this function using two multipliers which will alternately be performing the two equations specified. Figure 9 shows a computation graph associated with the description given in Figure 8. Horizontal lines labelled t_1, t_2, t_3 and t_4 represent word boundaries which occur when the input data is valid and every n bit times thereafter until new input data is valid, where n is the number of bits in a word; thus,

```
FOR I = 1 TO 20 DO

    z = c[i]*x(n) + d[i]*x(n-1)

y(n,i) = a[i]*y(n-1,i) + b[i]*y(n-2,i) - z
```

Figure 8. Description of a second order section filter bank.

moving down in the figure corresponds to the passage of time. The boxes representing functional units and delays have a length approximately equal to the latency of the unit, i.e., the amount of time between the entrance of a least significant bit of a data word and its emergence at the output of the functional unit. Connections should be regarded as taking zero time. To make the graph more readable, actual latency times have been distorted and connections have time greater than zero. One significant point is that the serial multiplier functional unit will have latency greater than a word time. To obtain maximum utilization of resources, the computations performed during the word time starting at t_3 can be done in parallel with the first two multiplications, resulting in the movement of the dashed computations in Figure 9 to the word time bounded by t_1 and t_2. The same computations are shown solid in that word time rather than dashed.

The minimum number of functional units needed to implement the description of Figure 8 is 2 multipliers, 2 adders and one subtracter. An assignment of particular functions to functional units is shown in Figure 9. However, to meet restriction (2) above and still allow the data to be pipelined through the multiplier and adders, we use an extra adder (an inexpensive resource). Adder 1 is now in use from boundary t_1 to boundary t_3, despite the fact that the actual use of adder 1 is half that time. During the remaining time, adder 1 is computing junk. The reason for this inefficiency is the requirement that the connections to adder 1 must be established at boundary t_1 and must be maintained until boundary t_3, i.e., for the entire operational time of adder 1. The resulting simplification of connection switches appears to result in a net savings of silicon area for most applications. Expensive functional units will be required to begin operation at word boundaries; this restriction permits expensive functional units to be reused one word time later, i.e., as soon as possible. Thus, there is no inefficiency in the use of expensive functional units.

Figure 10 shows an implementation of the example shown in Figures 8 and 9. Only the connections that may change are shown; fixed connections have been eliminated to simplify the figure.

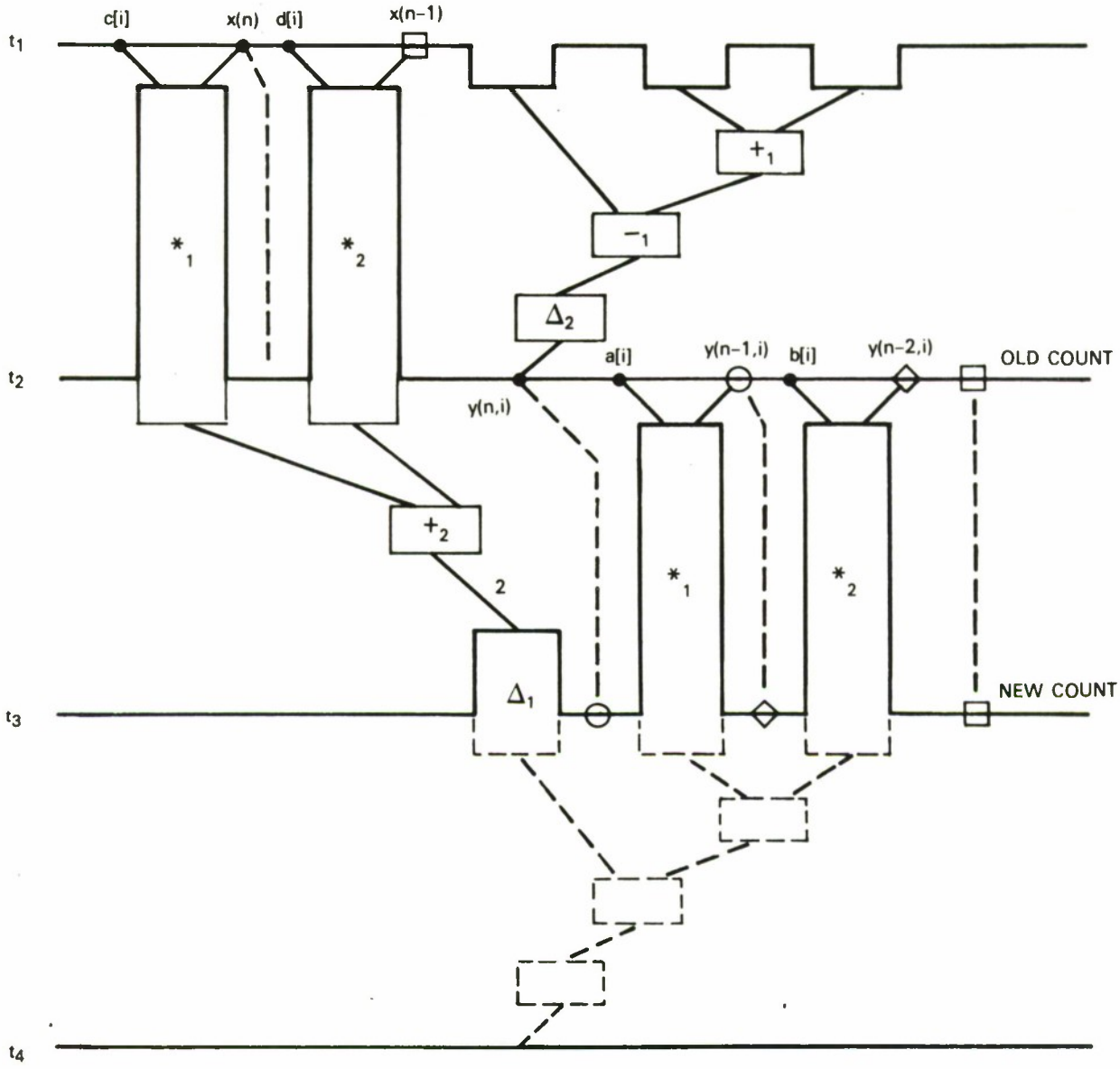
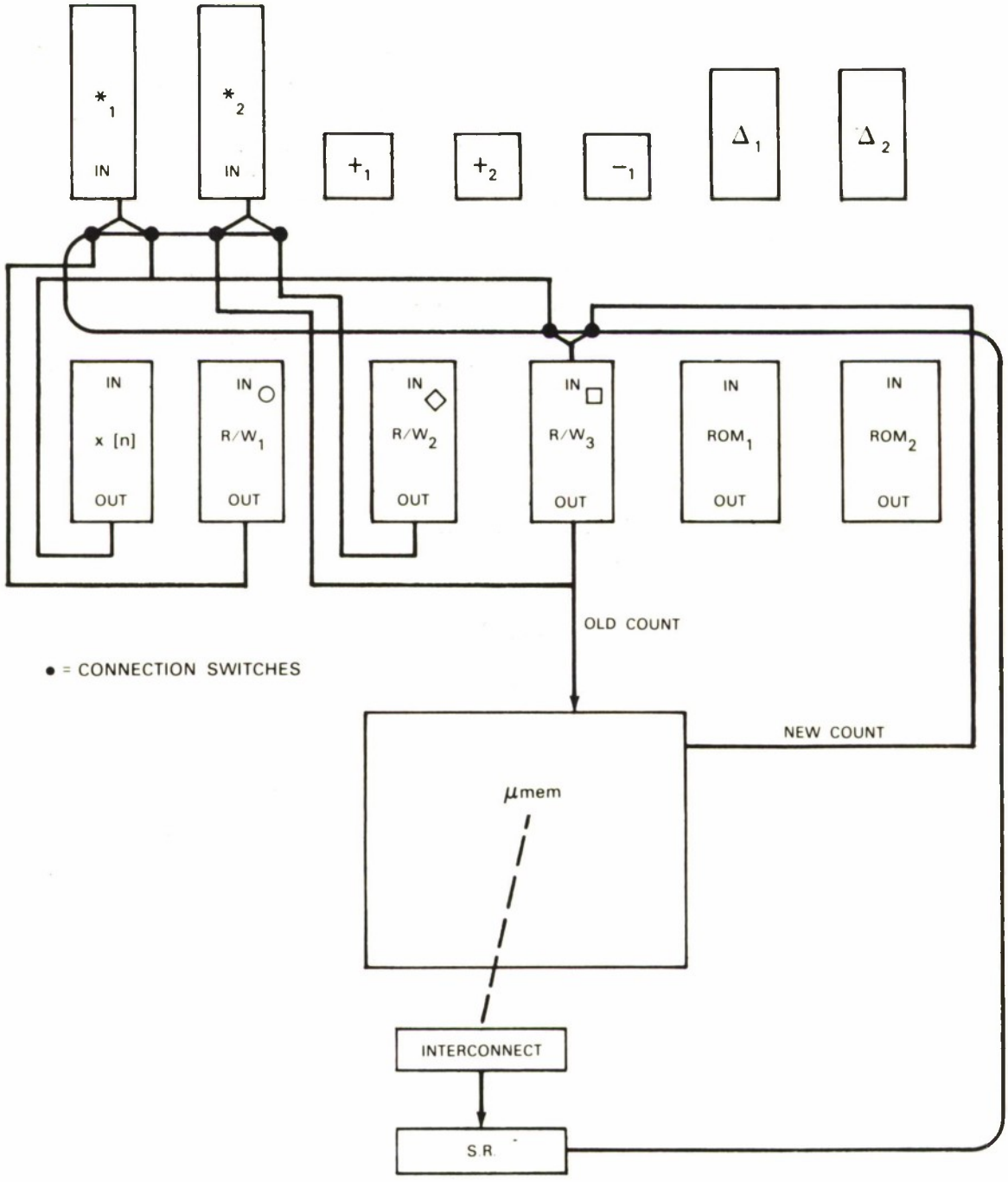


Figure 9. Example description and computation graph.



153357-N

Figure 10. Example implementation.

Algorithms and Status

Several algorithms are needed to translate the signal processing language description of the hardware to be designed into silicon. In particular, algorithms will be required to perform the following functions:

(1) Parse signal processing language.

(2) Schedule functional units into microinstructions.

(3) Delay equalize the scheduled design.

(4) Group the storage registers into read/write and read-only storage units.

(5) Assign functions to units.

(6) Generate microcode.

(7) Complete the physical design, e.g., layout, routing.

Phase I currently uses algorithms of types (1) through (3) and algorithm (7); algorithms (4) through (6) are unique to Phase II. The Phase I algorithms will have to be modified somewhat due to the addition of explicit loops and redundant inexpensive resources. Also, a control unit layout with parameterization must be completed, new primitives for storage units and connection switches must be generated, and a modified overall automatic layout method must be determined.

Algorithm (4), which was described earlier within the section on interconnection, has been designed but not programmed. The purpose of algorithm (5) is to assign actual functional units to functions in such a way as to minimize the number of switch connections necessary. Algorithm (6) involves the generation of necessary sequencing instructions, insertion of redundant microwords to maintain efficient data flow in the presence of loops and conditionals, and specification of the contents of the next address field in the microword. Algorithms (5) and (6) are currently being developed.

III. APPLICATIONS

A. DTW WAFER SYSTEM

1. Simulations

A bit-level simulation of the DTW level building algorithm was used for architectural studies concerning the wafer-scale implementation of a DTW level building processor. The simulation was written in the language "C" on a VAX-11/780. Much of the performance evaluation was done with a data base provided by the University of California/Berkeley containing both isolated-word and connected-digit utterances.

The DTW processor is composed of an array of computational elements (1 × N) each with nearest-neighbor communication. Each element performs a distance measurement between a frame of a stored reference utterance and a frame of an unknown test utterance. It then adds that computed local distance to the minimum accumulated path of the previous iteration from itself and the elements nearest it.

The issues which were studied in detail and the resulting parameters selected are as follows:

(1) Filter bank coefficient quantization — 6 bits;

(2) Distance metric — 4-bit approximation to 6-bit squared Euclidean;

(3) Path constraint and path weightings — flipped Itakura with weights 1/2, 1, and 2 applied to the different paths;

(4) Local distance accumulation quantization — 12-bit, saturating arithmetic.

Coefficient Quantization

Though the actual knee of the error curve occurs at 4-bit quantization of the filter bank coefficients, additional improvement occurs out to 6 bits. For some speakers this improvement can be considerable. Because a compact approximation to a 6-bit square can be realized, and because of the performance improvement, a 6-bit coefficient quantization was selected.

Distance Metric and Approximations

The squared Euclidean distance metric proved to be superior to the others considered. Approximations to this distance metric which would increase its dynamic range at reasonable levels of integration complexity were considered. The approximations concerned mainly the squaring operations, although others could have been attempted. The 6-bit square approximation is accomplished using a 4-bit squaring table. The 6-bit magnitude difference of individual frame coefficients are separated into two overlapping 4-bit fields. When the magnitude

of the value to be squared is greater than that which can be represented by the 4 LSBs, the upper 4 bits of the 6-bit value are entered into the squaring circuit. This constitutes a division by 4 (2-bit right shift). Subsequently, a multiplication by 16 (4-bit left shift) is required on the output of this squaring operation. A simple rounding mechanism is provided. If the MSB of the 2 unused bits of the input is set, then the LSB of the upper 4-bit quantity is set. The scaling of the output is implemented using multiplexers on the output register of the squaring circuit. The square approximation works as well as the exact implementation in our simulations. Even more simplified approaches which produced greater error in the squaring operation, produced similar results.

Path Constraint and Weighting

The "flipped" Itakura path constraint was developed particularly for the row-oriented architecture. This path constraint uses only one row of look-back in its path minimization process. This eliminates the need to include extra storage registers for delaying data by more than one iteration. Both the "flipped" Itakura path constraint and its weighting were selected for their demonstrated recognition performance advantages. However, it appears that this advantage diminishes with increased range in the local distance calculation resulting from either increased coefficient range or by the metric selected. When 4 bits are used to represent the channel energies, percentage error rates vary by as much as 38% depending on the constraint used; while at 6 bits, the variation is only 11% (using squared Euclidean).

The path weights specify the scaling applied to the local distance before it is added to the minimum path value. This scaling is applied to the newly computed path distance before it is passed to the next nodes. Weights of 1/2, and 1 and 2 are applied to paths with shifts of 0, 1 and 2 units respectively.

Path Accumulation

Tests were conducted on the quantization of the local distance accumulation. Preliminary statistics on average local distances along selected global warping paths showed these values to be around the 6- to 10-bit range. Tests on quantization with saturation showed that there was no decrease in error rate after 10 bits of quantization. Twelve-bit accumulation was selected, first to maintain a reasonable ripple carry propagation time in the accumulator, and second, to prevent pruning of valid warping paths by early saturation.

2. DTW Architecture

The DTW wafer is a pattern matching processor composed of a linear array of computational elements. Each element performs a distance calculation and a warping-path calculation. These operations define the two basic cell types used in composing an element in the array. The distance calculator (DCALC) contains the circuitry necessary to perform an accumulation of the square of the magnitude difference of two 6-bit coefficients, the squared

Euclidean distance metric. It also contains the buffer which holds a frame of the input speech for which we are trying to find a match. The path calculator (PCALC) contains path selection logic (minimizer) and adders for adding to the selected global path value the distance between frames at this node. Also contained in the PCALC is the level processor which performs inter-reference word decision processes. The level processor maintains the best global path across the reference word set out to each frame of the input speech. Initialization information and results are also loaded and unloaded through the level processor registers. The detailed architecture of the cells was determined from the simulations.

DCALC

The distance calculator first computes an approximation to the square of the difference of the coefficients. The two 6-bit coefficients are subtracted, and the two's complement of the result is taken if the sign bit is high. The 6-bit magnitude difference is entered into a 4-bit squaring unit as follows: the least significant 4 bits are entered if the value is representable by 4 bits, otherwise the four most significant bits are entered. In the latter case, the output of the square circuit is placed in the upper 8 bits of a 12-bit output register. This rescales the output by the square of the inverse of the scaling applied to the input.

The output of the squaring circuit is fed into an accumulation loop. The accumulator is a 12-bit, magnitude-only, saturating adder with ripple carry. The augend is either the partial accumulation or, for reinitializing the accumulation for the next frame, zero. At the end of a frame computation, the accumulated local distance is loaded into a 12-bit, parallel/serial conversion register. The output of this register is connected to the path calculator.

Inputs to the distance calculator come from two sources. The reference data is provided on a 6-bit bus, which is routed globally to all active elements. The input data (or unknown) is held in a 6×12-bit buffer in the distance calculator, called the I-buffer. The I-buffer also retains the initial frame data for each level. This double buffering scheme allows for reference word changes (level reinitialization) without reloading input data, thus providing 100% processor utilization during the matching process.

PCALC

The PCALC performs both local path decision processes and level decision processes. During each reference word, the path processor compares all incoming paths from the previous iteration. The smallest path is identified and is used to produce the outgoing path which has added to it the local distance computed at this node. This process continues, producing the global warping paths which best match the current reference word out to each associated frame of the input speech spanned by the processing array.

In the level processor these final path values are compared to the minimum global paths to each node in the array for all previous reference words. When the current reference word is longer (in frames) than the previous word, then the resident level information

must be shifted to align it with the resulting paths for the current word. The shifted data in the earliest input frame alignment positions are transferred to an external memory. This implies a preordering of the reference vocabulary to insure that the words enter in a shortest-to-longest fashion. This ordering is a one-time operation performed off-wafer and does not affect the wafer performance. This transfer operation is performed by concatenating three internal registers of the level processors into three long, serial-bit streams and shifting their contents out through three output pins. These registers contain the path accumulation, column traceback, and reference word pointer and are each 24 bits in length.

One failure-avoidance feature was added to prevent a single element dropout from rendering the entire wafer inoperative. This feature is an enable bit which is included in each element. The enable bits are linked into a long shift register. Cells are enabled by entering a serial bit pattern equal in length to the number of linked elements. The pattern contains a one in the position of each active element. Disabled cells are bypassed by actively switching the inputs of the cell directly to the outputs.

3. Cell Designs

Cell and wafer layout for the DTW wafer-scale processor has continued during the period. Of the two wafer cell types, the DCALC and PCALC cells, the DCALC was the first to be laid out. The internal logic of this cell has completed layout, and the entire cell, as well as time critical subportions of the cell, has been submitted to MOSIS for fabrication (see Table I). The PCALC cell has been floor-planned and layout is proceeding.

TABLE III-1 [U]

Mosis Run	Projects Submitted
Feb. 21 — CBPM2	I-buffer
Mar. 14 — CBPE2	I-buffer (revised) DLA cell (Squaring Circuit) Diff/Absv.
Apr. 4 — CBPM2	DLA cell (revised) Accumulator Complete DCALC

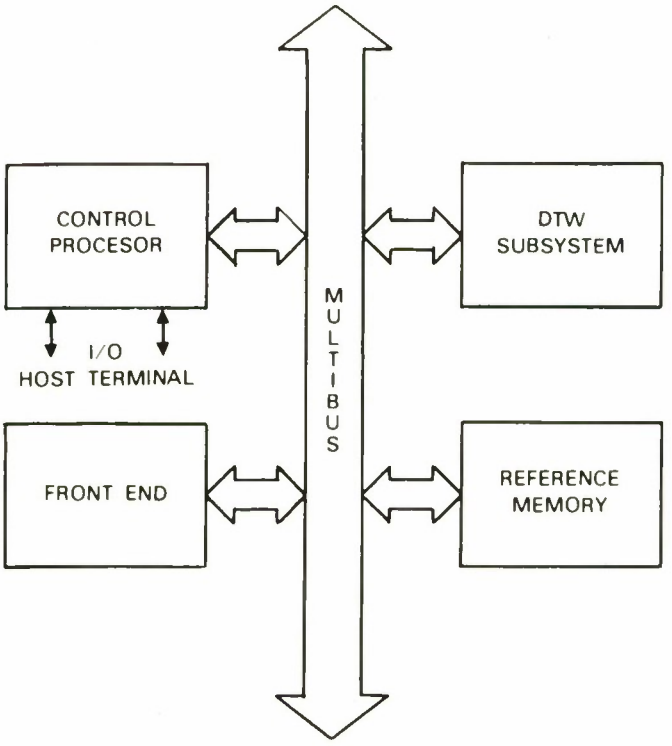
Wafer floor planning and signal-routing channel allocation have also proceeded in the last Quarter. The wafer will consist of vertical columns of fabricated DCALC, PCALC pairs. The row architecture of the completed system will be realized by routing nearest-neighbor connections in the routing channel between the pairs. The columns will be connected together alternately at the top and bottom of the wafer, thereby "snaking" the row

up and down through the wafer. Broadcast data and control signals will enter the top of the wafer and be distributed in the routing channels on the outside of the DCALC and PCALC pairs.

4. Support System

A compact, peripheral hardware unit is being developed to support the DTW wafer application. This system will contain the necessary processing resources to parameterize and recognize continuous speech in real time with a vocabulary of up to 4000 words. As shown in Figure 13, the support system includes: (1) a general purpose control processor, (2) a DTW subsystem containing the wafer, and (3) a front end having speech analysis capability. Each system block of Figure 13 will be one or more Multibus boards.

The control processor coordinates all data traffic over the Multibus, including (1) the accepting of processed digitized speech from the front end, (2) DTW subsystem configuration, high level control, and interlevel speech data processing, and (3) data formatting for display of recognition results at an operator terminal.



153345-N

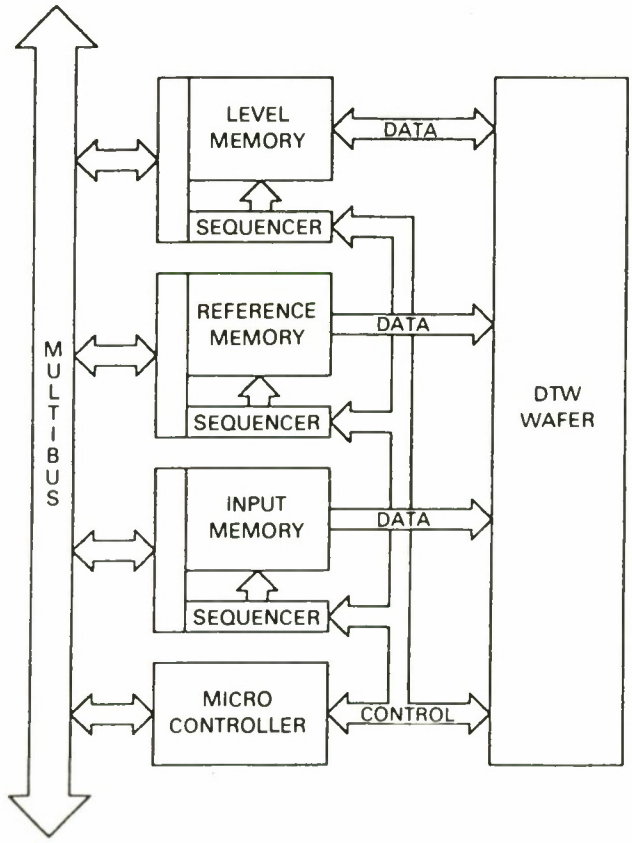
Figure 13. DTW support system block diagram.

Control Processor

A Heurikon HK68 single board microcomputer has been selected for use as the high-level, DTW system control processor. The HK68 is a 68000-based Multibus board with 4 Mbyte/sec quad channel DMA, 1 Mbyte of dynamic RAM, 64 kbyte EPROM capacity, 4 serial I/O ports, 3 programmable timers, vectored interrupts, and a number of other features. Application software for the HK68 will be developed using a Hewlett Packard 64000 microprocessor development system. The software will be written using a combination of a 68000 assembler and a "C" compiler.

DTW Subsystem

The DTW subsystem is projected as a slave controller board to the HK68 control processor, consisting of: (1) the DTW wafer, (2) three subsystem memories, and (3) a microcontroller for coordinating fast control and wafer-to-memory data transfers. A block diagram is shown in Figure 14.



153346-N

Figure 14. DTW subsystem block diagram.

The current design includes 8 kbytes of input memory, 1 Mbyte of reference memory, and 8 kbytes of level memory. The reference memory, currently planned as thirty-two 64 K × 4 DRAMs, may be expanded to 2 Mbytes. It will be a dual port memory and will be constructed on a separate card. Each memory is accessible via Multibus or a fast address sequencer under the control of a 2910A-based microcontroller. Multibus access is also provided for sequencer address initialization, for downloading microcode to the microcontroller writable control store, and for initiation of microcontroller program execution. Special 3-bit data formatting is provided at the level memory/wafer interface. A subsystem clock generator will provide nominal 4 MHz and 8 MHz two-phase, nonoverlapping clocks to the wafer at appropriate CMOS voltage levels, together with a FRAME marker and a family of useful clocks for other portions of the subsystem. The microcontroller will be able to generate a Multibus interrupt to the control processor at the conclusion of each subsystem level processing interval.

The general notion of a microcontroller-based DTW subsystem with Multibus interface will provide the flexibility to make control timing adjustments or to change subsystem memory contents throughout the wafer development process. The first planned use for the support system is to control a functional segment of the full wafer array comprised of individually-packaged PCALC and DCALC cells.

IV. TESTING

A. OPTICAL PROBE DISTRIBUTION

An optical probe is now in operation at the University of Utah. Laser sources and manuals have been delivered to the National Security Agency; Stanford University; Mississippi State; Utah; Bolt Beranek and Newman, Inc.; and the Information Sciences Institute. Manuals have been sent to Columbia and North Carolina, and a second lot of 10 laser sources is now being fabricated by an outside vendor.

V. REDUCED-DIMENSION LASER LINKS

It has been shown that laser links as small as 9×9 μm^2, a factor of 4 smaller in area than the present design, can be programmed with better than 99.9% yield. In fact, the programming seems to become *more* reliable as the size decreases, since the power threshold for making a substrate contact (which puts an upper limit on laser power) increases. The results are for the simple pyramid pattern, which lacks any polyimide or thick glass insulator. A test pattern for reduced-dimension, complete RVLSI links is being designed.

REFERENCES

1. Restructurable VLSI Program Semiannual Technical Summary, Lincoln Laboratory, M.I.T. (31 March 1984), DTIC AD-A148145.

2. L.E. Leiserson, F.M. Rose, J.B. Saxe, "Optimizing Synchronous Circuitry by Retiming," Third Caltech Conference on VLSI, 87-116 (March 1983).

3. E.L. Lawler, *Combinational Optimization: Networks and Matroid*, Holt, Rinehart and Winston, NY, 74 (1976).

4. K.W. Crouch, "L5 User's Guide," Project Report RVLSI-5, Lincoln Laboratory, M.I.T. (7 March 1984).

5. R.R. Lyon, "Two's Complement Pipeline Multipliers," IEEE Trans. Commun., **COM-24**, 418-425 (1976).

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-85-206	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Restructurable VLSI Program		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Summary 1 October 1984 — 31 March 1985 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR*(s)* Gerald C. O'Leary		8. CONTRACT OR GRANT NUMBER*(s)* F19628-85-C-0002
9. PERFORMING ORGANIZATION NAME AND ADDRESS Lincoln Laboratory, M.I.T. P.O. Box 73 Lexington, MA 02173-0073		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order 3797 Program Element No.61101E Project No.3D30
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE 31 March 1985 13. NUMBER OF PAGES 44
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Electronic Systems Division Hanscom AFB, MA 01731		15. SECURITY CLASS. *(of this Report)* Unclassified 15a. DECLASSIFICATION DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT *(of this Report)* Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*		
18. SUPPLEMENTARY NOTES None		
19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)* VLSI Restructurable VLSI (RVLSI) programmable interconnect defect avoidance customization hardware description language placement routing systolic array integrator waver-scale systems speech recognition		
20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)* This report describes work performed on the Restructurable VLSI Program sponsored by the Information Processing Techniques Office of the Defense Advanced Research Projects Agency during the period 1 October 1984 through 31 March 1985.		