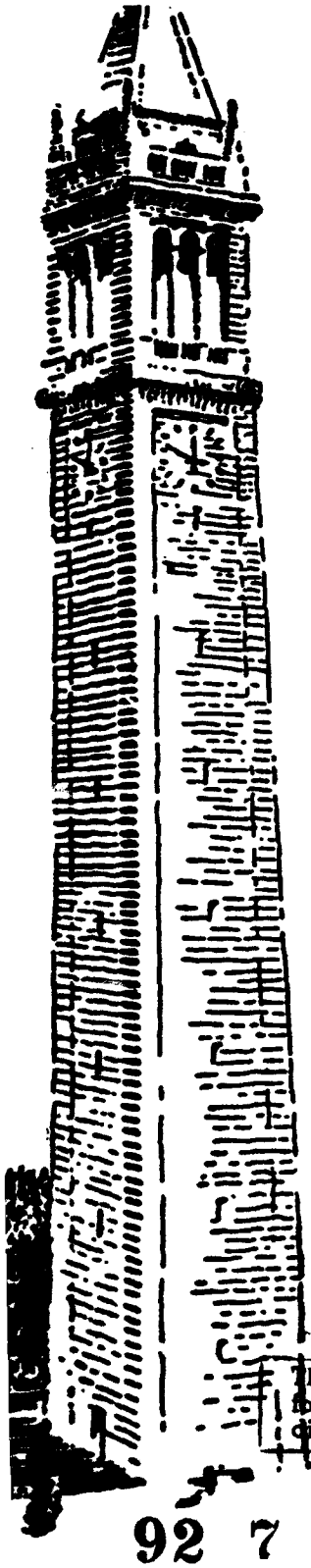
DESIGN and PROTOTYPING OF HARD REAL TIME SYSTEMS



Electronics Research Laboratory
University of California
Berkeley, CA 94720

Final Report
June 1990

Principal Investigators

R.W. Brodersen
rwb@zabriskie.berkeley.edu
(415) 642-1779

A.R. Newton
A.L. Sangiovanni-Vincentelli
J.M. Rabaey
R.K. Brayton
E.A. Lee
D.G. Messerschmitt

SPONSORED BY
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

MONITORED BY
SPACE AND NAVAL WARFARE SYSTEMS COMMAND
UNDER CONTRACT No. N00039-88-C-0292 (TASK 5)

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley, CA 94720

Table of Contents

1. CAD INFRASTRUCTURE AND TOOLS

1.1. Massively Parallel Algorithms for Three-Dimensional Device Simulation (A. Sangiovanni-Vincentelli)

Recent advances in processing technology create the need of modeling physical phenomena described in terms of three-dimensional geometries. For instance, submicron technology for MOS devices requires the accurate modeling of narrow-channel effects which, in turn, imply the use of 3-D discretization. Three-dimensional effects must be modeled also in advanced bipolar structures. However, including these new effects yields an enormous increase in CPU time if the present 2-D algorithms are extended to cover the three-dimensional case on a conventional processor. Large vector supercomputers have been used to simulate this class of problems: [1] a three-dimensional device simulator was developed based on a seven-point finite-difference discretization and the biconjugate gradient method was used to solve the non-symmetric linear system arising from the previous discretization. The vectorization provided a speed-up of sixteen for typical problems. Since supercomputers offer such a limited degree of parallelism, other approaches are under investigation to efficiently solve very large simulations. Recently, massively parallel algorithms have been used for linear capacitance evaluation in three-dimensional structures showing good computational performance [2]. In this work we present a full 3D device simulator developed on a Connection Machine [3]. The CM is a massively parallel SIMD computer with up to 65,536 bit serial processors. Each processor has 64 Kbits of memory. Communication is either a fixed distance on an N-dimensional grid, or direct to an arbitrary processor based on a hypercube.

In this paper we will extend the techniques presented in [2] to device simulation.

The simulator consists of a Poisson solver and of a general drift-diffusion equation solver. Poisson's equation is discretized on a three-dimensional finite-difference grid, with non uniform spacing which has been mapped on the CM architecture by allocating a processor to each grid node. The non-linear equations have been solved by Newton iteration. The linear system which has to be inverted at each non-linear loop is symmetric, positive definite and diagonally dominant. These properties guarantee that Incomplete Cholesky Conjugate Gradient method converges to the solution. A red/black partitioning [2] of the unknowns has been performed to achieve a high parallelism. The parallelism of this method has been enhanced by exploiting the

processors which are idle during the preconditioning phase. More specifically, defining the jacobian matrix $B = \left[\tilde{LU}\right]_{r/b} + E_{r/b}$, *where* $\left[\tilde{LU}\right]_{r/b}$ factorization of B for a proper r/b ordering, we compute also the preconditioning for the associated b/r ordering. It is possible to show that this technique reduces the maximum value of the entries of the error matrix E. This preconditioning does not requires any additional overhead, because the two solutions can be evaluated using the idle processors during the preconditioning phase. Table 1 shows the improvements in terms of iterations for the new method when compared with the standard one. The test structure is a 3D MOS capacitor on a uniform substrate ($Na = 10^{16}$) and the number of grid nodes was 3120.

When the current flow is not negligible, the solution of the full set of drift-diffusion equations is required. The Scharfetter-Gummel approach [4] is used to discretize the continuity equations, the coupled Newton method to solve the non-linear system of algebraic equations, and the Biconjugate Gradient method [1] to invert the jacobian matrix. Since now there are three variables for each grid node, as opposed to one for Poisson's equation, we use a two-level matrix, where the higher-level accounts for the interactions among adjacent nodes, while the (3x3) lower-level matrices are used to represent the interactions among the variables associated with the same node. Following this approach, the Biconjugate Gradient method has been implemented replacing each arithmetic operation by its matrix counterpart. The preconditioning scheme has been formulated using a red/black partitioning of the high-level matrix. While alternative preconditioning approaches are now under evaluation, it is useful to consider the performance of the overall method for a simple diode polarized in high-injection regime. The diode has $N_d = N_a = 10^{-16}\ (cm^{-3})$, the whole domain is a cube of 3μ of side, and the n-doping profile is modeled as a 1μ cube. The number of iterations required by the Newton method to reach convergence and the global number of iterations required by the BCG technique are shown in Table 2 as a function of the applied voltage and of the number of grid nodes. The initialization of the variables has been performed by a linear extrapolation from the two previous solutions. An important parameter to estimate the accuracy of the solution is given by the conservation of the current even at very low levels. We have simulated the same diode in reverse region (1V), evaluating the currents generated in the depleted region. The current conservation was achieved up to 5 digits for a current density of the order of 10^{-16} Ampere/μ^2 using a relative convergence tolerance of 10^{-6} for the Newton method and 10^{-8} for the BCG. Our preliminary measurements lead us to expect a performance of 700 Mflops on a fully configured Connection Machine.

Bias Step	Non-lin. It.	I_{st}	I_{New}
0.0	9	202	155
0.1	3	63	47
0.2	3	55	45
0.3	3	60	47

Table 1: Comparison of the standard iteration with the new one.

1.2. Almost Periodic Fourier Transform

Many circuits of interest have nonlinear elements and have inputs of incommensurable frequencies. Simulating such systems with harmonic balance requires computation of an Almost Periodic Fourier Transform (APFT).

The APFT can be described as follows: Given is a set of n fundamental frequencies and $2n + 1$ times. Given the amplitudes and phases of the fundamentals it is easy to compute the function value at the times; this transform is called F^{-1}. Conversely, given the function values at the $2n + 1$ times, we can construct the amplitudes and phases; this transform is F. Both F and F^{-1} are linear maps. The difficulty is that if the times are not carefully chosen, the matrices representing F and F^{-1} are extremely ill-conditioned, resulting in large numerical errors.

The standard Fourier transform is a special case in which the frequencies are evenly spaced. In this case using evenly spaced timepoints gives a matrix whose condition number is $O(n)$. Choosing evenly spaced time points in the APFT case typically gives condition numbers on the order of $n!$.

The APFT problem then is the selection of a good set of timepoints. In previous work we found a good heuristic solution. We chose a larger number of timepoints, say $2(2n + 1)$. Each timepoint determines a row of F^{-1}. We iteratively add to the set of 'chosen' timepoints the unchosen one whose row is most nearly orthogonal to the rows corresponding to the chosen timepoints; the algorithm is similar to Gram-Schmidt orthogonalization. In practice this yields and F^{-1} whose rows are 'nearly orthogonal' in a well-defined sense, from which we prove that F and F^{-1} are well-conditioned.

We are currently seeking an algorithm which is deterministic (uses no randomness), efficient, and constructs a set of 2n+1 timepoints yielding an F whose condition number is less than a specified bound.

One piece of this puzzle is to compute an 'almost period' of the system: a value T whose residue modulo each of the given periods is smaller than a given limit ε. We have shown how to do this by applying the Lenstra, Lenstra, and Lovasz (LLL) basis reduction algorithm. The algorithm efficiently constructs a short (nearly shortest) vector in a lattice described by an arbitrary set of basis vectors; it may be thought of as a generalization of the continued fraction algorithm.

Ideally, we would like to choose a set of timepoints which give a matrix which is nearly identical to the matrix representing the standard Fourier transform. We suppose that the frequencies are described in a way which includes the linear relations between them. For instance a set of frequencies could be specified as $\{f_1, f_2, f_3, f_1, f_1 - f_2, \ldots\}$. If there is a map of the 'independent' frequencies (f_1, f_2, and f_3 in the example above) into the set $\{0,1, \ldots, n-1\}$ such that the full set of n frequencies maps 1-1 onto that set, then Kronecker's theorem proves that there is a time T_1 at which the phase angles of the fundamental frequencies are (to within an arbitrary degree of accuracy) the values $0, 2\frac{\pi}{n}, 4\frac{\pi}{n}, \ldots, \frac{(n-1)2\pi}{n}$. In this case the row of F^1 determined by T_1 is (approximately) a row of the standard Fourier transform matrix. Furthermore, if the full set of times is equally spaced, i.e., $T_1, 2T_1, \ldots, (2n+1)T_1$, then the APFT matrix F^{-1} is (approximately) the Fourier matrix. In this case it is not only well-conditioned, but furthermore its inverse F, which is the matrix that we actually need, is (approximately) known in advance. There are well-established numerical techniques to then compute the exact value of F from the

constructed F^{-1}.

This leaves us with the two fundamental problems on which we are now focusing.

First: does a 1-1 map from the set of frequencies to $\{0,\ldots,n-1\}$ exist? If so, how do we find it? And if not, what do we do next?

Second, how do we find the value T_1 whose existence is guaranteed by Kronecker's theorem?

The first question appears to be essentially combinatorial in nature, and we do not currently have a good approach to solving it.

The second question may be related to the problem of finding an almost period, which we have solved as described above. For example, in some cases using $T_1 = T/(2n+1)$ works. In other cases it is possible to reapply the LLL algorithm to compute an acceptable T_1. Thus there is some promise to this technique, though we do not know how to solve the general case.

1.3. Finite-Time Theory of Simulated Annealing on Special Energy Landscapes (A. Sangiovanni-Vincentelli)

The best theoretical results on simulated annealing apply to arbitrary spaces, including those for NP-hard problems, and are necessarily somewhat weak. In practice, the 'cooling schedules' used in annealing are more rapid than those analyzed by the theory, and the results obtained are much better than theory would predict. The strength of the practical results, combined with the fact that it is easy to construct problems on which annealing will not work well, means that problems encountered in practice have special properties of which annealing takes advantage. Thus we take a two-pronged approach: studying properties of energy landscapes of real-world problems; and analyzing the behavior of annealing on landscapes with given properties.

We have observed many fractal properties of combinatorial problems, focusing on placement problems. In particular, the sequence of energies observed over time is a Brownian or fractional Brownian motion (fBm). We have proved that a random walk on a fractal landscape produces such fBm 'energy trajectories', and we conjecture that this mechanism is responsible. Also, direct sampling of points in the landscape shows an approximate power-law relation between distance and energy difference. Such a relation is a natural extension of the definition of fractalness in Euclidean space (the landscapes of interest do not lie in Euclidean space).

Most recently we have studied a class of mathematically defined deterministic self-affine fractal energy landscapes on the reals. In this limited domain rigorous results can be derived using the tools of rapidly mixing Markov chains.

In particular it is proved that using a geometric cooling schedule, the expected energy difference between the state found by annealing and a true global minimum decreases approximately as a (negative) power of the total time spent annealing. The power itself depends on parameters of the problem instance.

In this 1-dimensional case, random sampling obeys a power law like that holding for annealing, and random sampling may even be faster (depending on the problem instance). But

for higher-dimensional fractals the speed of annealing is basically unchanged while that of random sampling decreases dramatically. The combinatorial spaces of practical interest do not have a well defined 'dimension' but do have many characteristics of very high-dimensional spaces, making these results particularly relevant.

These are dramatic new results. Previous formal studies of annealing have required that the time spent be exponential in the problem size, which is unrealistic. In this case the time is power-law in the 'quality' of result desired, and polynomial in D (for a D-dimensional fractal). Depending on whether range-limiting is employed, the time may be polynomial or exponential in the log size of one 'side' of the D-cube.

In short, the theoretical analysis performed yields strong results for the restricted (but we believe relevant) class to which it applies. It does so using a methodology which reflects what we think to be the way annealing functions in practice.

We hope to extend these results to random fractals, which will require developing a completely new set of mathematical tools. We also plan to use the theoretical analysis to guide further examination of practical problems, and ultimately to construct efficient annealing schedules and estimate their performance.

1.4. Implementation of the DFT-based Quasi-Periodic Steady State Analysis in Spectre (A. Sangiovanni-Vincentelli)

Spectre [5] is a harmonic balance simulator for analog and microwave circuits. It analyzes nonlinear analog circuits using the harmonic balance technique. Linear devices can be evaluated directly in the frequency domain. Nonlinear devices are in general impossible to evaluate in the frequency domain directly; therefore, the excitation spectra of a nonlinear device is first transformed into time domain. The response is then evaluated in the time domain and transformed back to the frequency domain. Currently, Spectre uses Discrete Fourier Transforms, and hence Fast Fourier Transforms, to convert periodic signals between time and frequency domain. For quasiperiodic signals, a special transform (the Almost Periodic FT [6,7]) is used to perform the conversions.

When the nonlinearities in the devices are algebraic, the coefficients of the sinusoids are frequency independent. Thus, for the purposes of evaluating the nonlinear devices, the actual fundamental frequencies are of no importance and can be chosen freely. In particular, the fundamental frequencies can be chosen to be multiples of some arbitrary frequency so that the resulting signals will be periodic, and DFT can be used. These artificially chosen fundamental frequencies are not actually used in the harmonic balance calculations. They are used only to determine in which order to place the terms in the spectra. In other words, we are only interested in the correspondence (mapping) between quasiperiodic and periodic harmonic indices.

In order to make computation involving quasiperiodic signals tractable, we need to truncate the frequencies into a finite set. The box and diamond truncation methods are two popular truncation methods. With the box truncation, only the first H harmonics of each fundamental frequency are considered. The diamond truncation limits the absolute sum of the indices of each fundamental frequency to be less than or equal to H. Currently Spectre only considers two

fundamental frequencies.

For a set of frequencies coming from a box truncation, we have $w \mid w = k1\,(\lambda 1) + k2(\lambda 2);\ 0 \le k1 \le H1,\ |k2| \le H2,\ k1 \ne 0 \text{ if } k2 < 0$ the correspondence between quasiperiodic and periodic harmonic indices for this particular truncation method is $k = (2H2 + 1)i\ k1 + k2$ (1)

For a set of frequencies obtained with a "diamond" truncation, $w \mid w = k1\,(\lambda 1) + k2\,(\lambda 2);\ |k1| + |k2| \le H, k1 + k2 >= 0, k1 != k2 \text{ if } k2 > 0$

$k = (H+1)\,k1 + Hk2$ (2)

is the mapping equation.

The users can specify a specific truncation scheme by specifying three parameters: H1, H2, and H, where

$|k1| < H1\ |k2| < H2\ |k1| + |k2| < H$

These constraints result in a combination of box and diamond truncation. In order to use the above equations to perform mapping from k1, k2 to k, we first expand the set of frequencies to its nearest box or diamond truncation whichever is smaller. Next, we used either equation (1) or (2) to map the quasiperiodic harmonic indices, k1 and k2 into their corresponding periodic indices k, and construct the spectra. In order to use FFT, the size of the spectra has to be a power of 2. Therefore, sometimes we would need to extend the size of spectra further to meet the requirement.

The DFT based spectral analysis has been implemented in Spectre, using the mapping technique described above. A few simple bench mark circuits were used to test the implementation. A sample circuit which contains a simple polynomial conductor shows the following result:

polynomial conductor: $i = v^4 + 2v^5 + v\,6$

No. of harmonics (diamond truncation)	7	6	5	4	3	2
FFT Time(sec)	0.43	0.443	0.15	0.17	30.05	0.03
APFT Time(sec)	29.52	12.73	4.95	1.58	0.58	0.42

The DFT (FFT) based spectral analysis is faster compared to the APFT based spectral analysis. The amount of speed up increases as the number of harmonics increases.

1.5. Generation of Analytical Models for Layout Interconnects (A. Sangiovanni-Vincentelli)

As the scale of integration and speed requirements of integrated circuits increase, accurate modelling of both on-chip and off-chip interconnects becomes necessary. As the widths of interconnection lines are scaled down, the thickness is either kept constant or scaled by a much smaller factor to limit line resistance. As a result of this, the fringing and sidewall effects dominate the interconnect capacitances, causing gross error in estimation, if parallel-plate approximation is used to compute line-to-ground and crossover capacitances. Also, shrinking widths of lines and increasing ratio of thickness to separation between lines increase tremendously the coupling capacitance between two adjacent parallel lines as a fraction of the total capacitance of each line. This makes estimation of coupling capacitances between adjacent lines indispensable for today's VLSIs.

Since the time constants associated with interconnects scale by a much smaller factor than those of devices, they are becoming increasingly significant in determining the speed of digital systems. At high speeds of operation, some of the interconnects act like transmission lines, and matching considerations become important in the design process. Moreover, when very aggressive design rules are employed, stray coupling can cause the logic state of a line to change due to switching of adjacent lines. In analog and mixed analog-digital circuits, needless to mention, performance can be extremely sensitive to interconnect parameters. All these factors combined have made interconnect modelling an active area of research.

Analytical expressions are available for modelling some simple configurations. However, one may encounter very complicated configurations in the layout for which closed form expressions are not known for modelling purposes. Numerical methods based on finite-difference, finite-element or integral-equation techniques can be employed to extract the desired electrical parameters (lumped or distributed). These methods solve Maxwell's equations in some form, and usually incur very high computational cost. Hence, there is a need for analytical expressions which can be used by CAD tools for fast extraction of large layouts. Such expressions will also provide useful insight to circuit and layout designers.

An experimental program has been developed which can generate analytical models for capacitances (line-to-ground and line-to-line) in some specific configurations by performing a series of accurate numerical simulations. The configurations considered right now are (a) single line, (b) adjacent parallel lines, and (c) crossing lines. The models are being used in the routing of analog circuits.

1.6. Analog Testing

In this project we aim to reduce production testing time by designing efficient test sets. As is done in digital testing, the test sets we design will be intended to detect the faults that are likely to happen in practice. Therefore, we base our test set design on a fault model developed for estimating yield in fabrication, where faults are characterized as either catastrophic or parametric. It has been shown in previous work that catastrophic faults are relatively easy to detect in analog circuits. Typically a few ac and dc tests are required. Consequently, most of production testing

time is spent attempting to detect parametric faults by verifying all of a circuit's specifications. As a result, any major reduction in testing time should come from reducing the number of specification tests that need to be performed. In addition, since testing is terminated upon the first failure of a test, an optimal ordering of tests can further reduce testing time.

To identify unnecessary specification tests and optimally order the remaining tests, we begin with a statistical description of parameter fluctuations in fabrication. We add specification tests to the set of optimally ordered tests one by one by looking at fault coverages of various subsets of tests. As tests are added, the current fault coverage of the set is determined, and the test set is complete when sufficient fault coverage has been reached.

In the last six months an algorithm has been implemented which optimally orders necessary specification tests and eliminates unnecessary specification tests. It has been tested on some examples and preliminary results show that we are able to attain approximately a 10 times speed-up over crude Monte Carlo methods for the same accuracy. The example below is an op amp which has 15 specifications and a process model containing 13 parameters. As can be seen in this example, very few specifications are critical. Future work needs to be done on improving the reliability of the algorithm, and more examples need to be tested.

SPECIFICATION (in order of importance)	FAULT COVERAGE OF CURRENT TEST SET	TEST YIELD
phase margin > 60 degrees	0.990224	0.739471
settling time (1V step, 0.1% interval) < 500ns	0.999894	0.9929
minimum output swing < -1.2 V	0.999939	0.99996
gain > 10000	0.999998	0.999907
maximum output swing > 1.2 V	1.0	0.999968
unity gain bandwidth > 4 MHz	1.0	1.0
maximum common mode input voltage > 1.0 V	1.0	1.0
CMRR > 80 dB	1.0	1.0
power dissipation < 1.1 mW	1.0	1.0
PSRR @ dc > 70 dB	1.0	1.0
PSRR @ 1kHz > 40 dB	1.0	1.0
minimum common mode input voltage < -1.0 V	1.0	1.0
slew rate > 2.5 V/us	1.0	1.0
maximum systematic offset < 0.001 V	1.0	1.0
minimum systematic offset > -0.001 V	1.0	1.0

* Cadence Design Systems

1.7. Constraint Generation for Routing Analog Circuits

The analog nature of real-world signals makes it impossible to eliminate completely analog circuits even when the host system is implemented in digital domain. As the trend for integrating both digital and analog functions continue, CAD tools become an essential part of design methodology. At this time, while CAD tools for digital design are well developed, tools for analog

design are still inadequate for improving design productivity. Also, at very high frequencies of operation, since digital technology has not matured, analog processing of signals is widely used. Although simulation and optimization tools are used by analog designers in the electrical design phase, one is yet to see use of general purpose CAD tools for layout design. Since performance of analog circuits can be very sensitive to stray effects, manual layout is time consuming and error-prone.

Layout of analog circuits is usually designed by designers in an iterative fashion. In each pass, after the layout is designed, all the layout parasitics are extracted and the circuit is simulated to check if performance specifications are met. If they are not met, possible trouble spots in the layout are "guessed," some changes made to the layout, and the process repeated. This is a highly inefficient approach, which can give rise to a large number of time consuming iterations.

We follow a novel *constraint-based* approach[8] for automating analog layout, both placement and routing, although presently we are concentrating on routing. Analog routing is considered as much more than a path-connection problem. In the first phase, critical layout parasitics are detected using sensitivity analysis[9]. Then a set of constraints is generated on these critical parasitics to ensure that circuit performance degradation remains within some specified limit. These constraints can then be used to drive the autorouter. This results in a high probability that performance specifications be met in the very first pass of routing. The parasitic constraints are of two types; *bounding constraints* and *matching constraints* . Since there can be many possible combinations of constraints which meet performance specifications, a unique algorithm is used to generate a set of constraints which maximize the flexibility of the router.

The program PARCAR (a parasitic constraint generator for analog routing) developed by us generates constraints on capacitances (both net-to-net and net-to-ground) for a given set of performance constraints. Interface programs have been developed for the simulators SPICE3 and SWAP to automatically obtain sensitivities with respect to all possible routing capacitances which may possibly exist in the layout. These sensitivities are fed to PARCAR for generating constraints. PARCAR is being interfaced to the area router ROAD and the channel router ROADRUNNER.

1.8. Logic Synthesis for Programmable Gate Arrays (Alberto Sangiovanni-Vincentelli)

Programmable gate arrays (PGA's) are becoming increasingly important architectures for rapid system prototyping. One common feature of such architectures is the presence of a repeated array of identical logic blocks. A logic block is a versatile configuration of logic elements which can be programmed by the user.

With the growing complexity of the logic circuits that can be packed on a PGA, it becomes necessary to have automatic tools that map logic functions onto these architectures. A simple application of the logic synthesis techniques used for semi-custom, cell-based architectures generally does not yield a satisfactory result and may be totally inadequate for such architectures. We are investigating the problem of combinational logic synthesis for two interesting and popular programmable gate array architectures: a RAM based architecture (Xilinx) and a multiplexor based one (Actel). We address the problem of synthesizing a set of Boolean equations using these

architectures such that minimum number of logic blocks are used.

In the Xilinx architecture, a basic block can implement any logic function of up to five variables. The algorithm has two main steps: decomposition and covering. Decomposition is applied on those functions which have more than five inputs. Various decomposition techniques are used to obtain a network in which all nodes have at most five fanins, e.g. classical decomposition techniques like Karp-Roth decomposition are investigated and compared with kernel extraction based on support of the functions. After decomposition, the new network may have some functions which can be realized by one basic block. So, covering methods are used to reduce the number of basic blocks needed for the network. We use exact covering formulation and also some heuristics to solve this otherwise intractable problem.

In the Actel architecture, the basic block is a configuration of 2-to-1 multiplexors. Our method of synthesis is similar to the one employed in MISII. However, guided by the architecture, we choose a different representation of subject-graph and pattern-graphs. We capture the gates in the library by a very small pattern-set. After covering the subject-graphs by pattern-graphs, we use an iterative-improvement phase which uses collapse and decompose operations to improve the result.

The results for both the architectures, as given in the following tables, are encouraging. Our results are given under column mis-pga. For Actel architecture, we compare our results with MISII results, whereas for Xilinx, we compare with results obtained from industry. Area is in terms of number of basic blocks.

XILINX

example	no. nodes	mis-pga	ind. results	
	(init.)	Area	time	Area
10bitreg	20	10	6.9	11
10count	126	23	154.2	19
180degc	146	21	74.5	34
3to8dmux	101	30	81.1	25
4-16dec	70	12	57.5	18
4cnt	66	17	35.0	10
8bappreg	141	27	64.9	27
8count	138	20	57.7	29
9bcasc2	135	34	1028.9	45
99bcasc2	133	29	91.9	42
arbiter	146	21	73.3	13

ACTEL

example		MISII		mis-pga
	Area	time	Area	time
f51m	52	21.5	50	59.7
bw	80	46.7	65	82.0
rot	310	108	3292	497.4
5xpl	52	19.5	46	43.2
c499	174	81.7	166	61.0
Cc1908	189	104.99	185	3258888.6
C5315	732	353	666	1865.3

It should be mentioned here that the algorithms described here are general: e.g. the algorithms for Xilinx architecture would work for any architecture in which the basic block implements a function of n variables, n >= 2.

As future work, we plan to look into minimizing the delay through the critical path. Extension to sequential circuits is also planned. It is hoped that such a tool will enable evaluations of different PGA architectures from synthesis point of view and result in design of better architectures.

1.9. Development of the SIS (Sequential Interactive Synthesis System) (R.K. Brayton)

Several critical issues regarding the internal data structures and the interchange format needed to be resolved. In particular, an extended version of BLIF (Berkeley Logic Interchange Format) was designed and is supported by SIS. (Done by S. Malik, Tzvi Ben-Tzur and K. J. Singh). This extended format permits specification of sequential elements (latches), timing information (clocking schemes) and state transition diagrams along with gates to represent digital logic. A reader and writer for this format was written and incorporated in SIS.

Area optimization using retiming and resynthesis ideas (developed earlier last year) was implemented as part of SIS. Surprisingly, the area improvements obtained using these techniques were negligible. This was shown to be a possible result of limitations of existing combinational logic optimization tools as well as being an inherent characteristic of the logical nature of some classes of circuits. These results were presented along with their analysis at HICSS 90 [10].

Performance optimization using retiming and resynthesis was satisfactorily tackled (work done by S. Malik and K. J. Singh). We considered the problem of redesigning a given pipelined circuit to meet a required cycle time. We demonstrate how the pipelined circuit can be transformed to a combinational circuit and show that solving a performance optimization problem for this combinational circuit is both necessary and sufficient for solving the performance optimization problem for the pipelined circuit. This is significant in two ways. Firstly, it shows that all known (as well as to be developed) ideas in combinational performance optimization can

be used in pipeline performance optimization. Second, it shows that these are enough, i.e., we need not consider special techniques for pipelined circuits.

1.10. Retiming and Initialization of Finite State Machines (R.K. Brayton)

Retiming is a general and powerful technique to perform delay or area optimization of sequential circuits. When sequential circuits are specified with an initial state, it is necessary to maintain the initial state across retimings. We have developed and implemented in SIS a simple and elegant method to perform this computation. If the initial state of the circuit is contained in a cycle of state transitions, our algorithm does not require any modification of the logic of the circuit, at the difference of previous approaches. For finite state machines, it is a simple matter during state assignment to make sure that the initial state is contained in a cycle of transitions if it is not already the case. This can be achieved at little or no cost in final circuit area. Once this condition is satisfied, initial states can be maintained across retimings with no need for costly state transition graph extractions, backtracking searches, or logic duplication.[11]

1.11. Technology Mapping for Area and Delay (R.K. Brayton)

Technology mapping is the process of mapping a logic network onto a set of predefined and characterized library gates. Most of the early work in this area has focused on minimizing circuit area, and only a small set of optimization techniques have been available for minimizing circuit delay. We have developed and implemented several new algorithms to perform delay minimization at a moderate cost in area, and we have obtained encouraging results (average speedups of 43% for a 14% increase in circuit area). These algorithms have been incorporated in the latest release of misII (version 2.2).

The basis of our approach is to decompose circuits in networks of trees, and use efficient algorithms to implement each tree separately. A similar decomposition was used in previous work (e.g. [12]). The main originality of our work is to incorporate within the same framework two kinds of trees: fanin trees, which implement the logic of the circuit, and fanout trees, which distribute the output signals of fanin trees to their destinations. In the present implementation, fanin tree optimization and fanout tree optimization are done independently. We are currently experimenting with the best way to integrate these two optimizations.

1.12. Optimum and Heuristic Algorithms for Finite State Machine Decomposition and Partitioning (A.R. Newton)

Techniques have been proposed in the past[15,16,17,18] for various types of finite state machine (FSM) decomposition that use the number of states or edges in the decomposed circuits as the cost function to be optimized. These measures are not reflective of the true logic

complexity of the decomposed circuits. These methods have been mainly heuristic in nature and offer limited guarantees as to the quality of the decomposition. In this work[14], optimum and heuristic algorithms for the general decomposition of FSMs have been developed such that the sum total of the number of product terms in the one-hot coded and logic minimized submachines is minimum or minimal. This cost function is much more reflective of the area of an optimally state-assigned and minimized submachine than the number of states/edges in the submachine. The problem of optimum two-way FSM decomposition has been formulated as one of symbolic-output partitioning and has been shown to be an easier problem than optimum state assignment. A procedure of constrained prime-implicant generation and covering has been described that represents an optimum FSM decomposition algorithm, under the specified cost function. Exact procedures are not viable for large problem instances. A novel iterative optimization strategy of symbolic-implicant expansion and reduction, modified from two-level Boolean minimizers, that represents a heuristic algorithm based on the exact procedure has also been developed. Reduction and expansion are performed on functions with symbolic, rather than binary-valued outputs. The heuristic procedure can be used for problems of any size. Preliminary experimental results have been presented that illustrate both the efficacy of the proposed algorithms and the validity of the selected cost function.

1.13. A Unified Approach to the Decomposition and Re-decomposition of FSMs (A.R. Newton)

A unified framework and associated algorithms for the optimal decomposition and re-decomposition of sequential machines have been developed. This framework allows for a uniform treatment of parallel, cascade and general decomposition topologies, operating at the State Transition Graph (STG) level, while targeting a cost function that is close to the eventual logic implementation.

Previous work in decomposition has targeted specific decomposition topologies[20,21,23]. In this paper[19], the decomposition problem has been formulated as one of implicant covering with associated constraints. An optimum covering corresponds to an optimum decomposition. It has been shown in this work that two-way or multi-way, parallel, cascade or general decomposition topologies can be targeted, simply by changing the constraints in the covering step. The relationship of this work to preserved partitions and covers, traditionally used in parallel and cascade decomposition has been indicated.

In many cases, an initial decomposition is specified as a starting point. Attempting to flatten a set of interacting circuits into a single lumped STG could require astronomical amounts of CPU time and memory. Memory and CPU time efficient re-decomposition algorithms that operate on distributed-style specifications and which are more global in nature than those presented in the past have been developed [22]. Arbitrary, interacting sequential circuits can be optimized for area and performance by iteratively applying re-decomposition algorithms across latch boundaries.

1.14. Testability Driven Decomposition of Large Finite State Machines (A.R. Newton)

The synthesis of controllers in the form of interacting finite state machines (FSMs) can result in improved performance and smaller area. In this work, we address sequential testability aspects in the decomposition of FSMs.

In this work, testability-driven decomposition techniques have been presented that realize controllers as FSM networks consisting of mutually interacting submachines which are fully testable for all single stuck-at faults without requiring direct access to the memory elements. Constrained decomposition techniques for the synthesis of fully and easily testable decomposed FSM networks have been described. Subsequently, an exhaustive classification of redundant faults that can occur in a single FSM embedded in a network has been presented. Associating each class of these redundant faults with a don't care set, an optimal decomposition technique has been described that synthesizes controllers as irredundant FSM networks with no area overhead by exploiting these don't cares optimally.

A FSM network can be flattened into a single State Graph and made fully testable using previously proposed synthesis techniques[24,25,. However, when dealing with a large, lumped FSM the don't care set under which the synthesis has to be optimal could be huge. Also, if the given initial representation is distributed, flattening it into a single lumped STG can require astronomical amounts of memory and CPU time.

It has been shown in this work that when a circuit is being implemented as an interconnection of FSMs, the required don't care set in an optimal synthesis procedure can be heavily pruned. Partitioning of logic serves to filter out the part of the don't care set which is not useful. The new synthesis procedure operates on a distributed-style representation of interacting STGs, which is considerably more compact than a lumped representation, carrying out a series of local analyses. This decomposition-based optimal synthesis technique is significantly more efficient, both in terms of memory and CPU time usage than previously proposed optimal synthesis techniques. It is, therefore, viable for circuits of greater size.

1.15. Cache Management Techniques for Multiprocessors, with an Emphasis on VLSI CAD Applications" (A.R. Newton)

General purpose multiple instruction stream, multiple data stream (MIMD) multiprocessors offer an attractive means of accelerating many computationally intensive tasks that arise in VLSI CAD systems; examples of such tasks include circuit simulation, logic simulation, placement, routing, design rule checking and fault simulation. MIMD multiprocessors are attractive because they are a general purpose solution that can also be exploited as high performance multiprogrammed computers. This is in contrast to special purpose CAD accelerators that only support single tasks such as logic simulation.

The particular class of shared-bus, shared-memory multiprocessors has gained commercial acceptance with offerings by Sequent, Encore and others. These machines offer high performance at reduced cost for applications with modest amounts of exploitable parallism. By efficiently supporting the shared memory programming paradigm at the hardware level, these

machines are usually much easier to program than those supporting message passing because the programmer does not have to worry about distributing data across multiple memories. Shared memory is especially attractive for programming CAD applications because it is often unclear how to efficiently map the problem at hand into a set of message passing processes.

The success of shared-bus multiprocessors is largely due to the efficient caching techniques made possible because of the shared bus [26]. Effective caching techniques are important in multiprocessor design because the provision of a cache at each processor masks the often severe access delay to main memory through an interconnection network. Unfortunately, computers with two or more caches require techniques for ensuring that the caches remain consistent: all changes to a piece of data must be reflected in all cached copies. It is now fairly well-understood how to enforce cache consistency in shared-bus multiprocessors by exploiting their broadcast capabilities [27]. Unfortunately, shared-bus architectures support relatively few processors (probably 50 or less), and it is unclear how to enforce consistency in more scaleable, non-bus architectures; this is the problem that we address in this research.

Cache consistency methods proposed to date may be divided into four classes: those in which shared writeable data is uncached, "snooping" protocols (for shared-bus systems), directory schemes, and software assisted techniques [28]; the first three classes are categorized as hardware based techniques. Hardware techniques enforce consistency in a manner that requires no special cache control instructions in program object code. Software techniques enforce consistency by inserting cache control instructions in object code at compile time. Since compilers generally have more information about the future behavior of a program than does the hardware at run time, software methods offer potentially better performance in terms of reduced average access time and network traffic. Several large, non-bus shared memory multiprocessor designs--the New York University Ultracomputer [29], IBM RP3 prototype [30], and University Illinois Cedar machine [31]--have dealt with the consistency problem using software methods. Only recently has work been reported on the investigation of hardware coherence methods for large non-bus machines [,32,33,34,35]. In most of these cases the methods proposed deal with specific architectures based on collections of buses connected in meshes, cubes or trees. In addition, few results have been reported characterizing the behaviour of parallel programs running on large numbers of processors; most machine designs have been based on analytic models.

This project is an investigation of the performance of cache consistency schemes suitable for large scale shared memory multiprocessors. It began with a comparison of three schemes using execution driven simulation of three parallel CAD programs. The first coherence scheme was simple: shared-writeable data is not cached. This scheme provided a reference point for the two others. The second scheme was a sectored variation of that published by Censier and Feautrier [36] in which tags are associated with each block of main memory to record all caches having copies; whenever a processor writes to shared-writeable data all other copies are invalidated using the data stored in the tags. While this method permits shared-writeable data to be cached, the additional memory required for tags and the additional network traffic required for invalidations may be excessive. The third scheme was novel in that it permitted the assignment of data to physical main memory locations to change dynamically. It required main memory to be distributed among the processors and handled it as if the distributed pieces were caches themselves. Since main memory is usually an order of magnitude larger than the aggregate cache memory, network traffic may be reduced if local memory hit rates improve and consistency traffic is low.

The comparison demonstrated three principal results. First, for the benchmarks considered the number of references to shared-writeable data is sufficiently high to justify the caching of shared-writeable data; significant reductions in average memory access time and network traffic were obtained for the second and third coherence schemes relative to the first. Second, the component of network traffic due to sharing and synchronization dominated the component due to normal cache misses. This showed that the reduction in normal miss traffic permitted by the larger effective cache size of the third scheme had little impact on improving performance. Third, the simulations showed that although the sectored version of Censier and Feautrier's scheme provided substantial performance improvement over the first scheme, the large block sizes required to minimize tag overhead introduced an excessive amount of false sharing.

The initial evaluation has been extended by considering the use of *tag caches* as a way to reduce tag overhead while minimizing false sharing. In a tag caching scheme, tags are not associated with each block of main memory but are stored in an associative memory indexed by block address. This permits a much smaller number of tags at the expense of a more complex main memory controller. Since the number of tags is much smaller than the number of main memory blocks in such a scheme, it is necessary to prematurely displace shared data from caches to get a free tag. preliminary results [37] on the effect of various tag cache sizes on network traffic and average access time show that Censier and Feaurier's directory method performs as well with a tag cache as with a full directory. Preliminary implementation considerations suggest that a tag cache can be implemented with the same low cost dynamic RAM as data memory.

This work is currently being extended by evaluating more realistic network models, including direct and indirect binary cubes, and cubes and trees of busses. Future work will also address the effectiveness of combining as a technique for reducing the significant performance degradation caused by synchronization. An initial evaluation of alternatives will use approximate queueing models, and will require the development of techniques that model synchronization. The most promising alternatives will then be evaluated using detailed simulations. This work will also be extended by acquiring more benchmark programs and by considering larger numbers of processors.

1.16. SLIP: System Level Interactive Partitioning (A.R. Newton)

Large electronic systems are usually constructed as a hierarchy of physical components, such as backplanes, boards, chip, and transistors. This hierarchy is typically defined as part of the system design process, for example as functionality is assigned to each chip. Alternately, an existing design might be reimplemented as a different hierarchy to take advantage of changes in VLSI technologies. For example, a TTL system could be reimplemented on a number of gate-array chips.

This problem poses conflicting demands upon algorithms which can be used to automatically generate system hierarchies. The problem size makes the speed and memory efficiency of the algorithms critical but the design must be modeled and optimized with enough detail to ensure that any constraints which are imposed by the packaging or system designer are satisfied.

We feel that this problem cannot be solved efficiently with a single algorithm. Rather, a sequence of algorithms which are optimized to perform specialized tasks must be used. The initial algorithms in the sequence partition and cluster the components of the system into a manageable number of tightly connected components. This will reduce the size of the problem so that a more careful algorithm can be used to assign clusters to packages and ensure that no package or user constraints are violated.

SLIP [38] is a framework for the development of algorithms that generate and improve system hierarchies. SLIP supports a representation of the hierarchy in the OCT [39] data model and provides a library of routines to maintain this representation as changes are made to the hierarchy. These routines simplify the implementation of partitioning algorithms and ensure the consistency of the representation as a sequence of algorithms are applied to the problem.

Status and Future Work

Recent work on SLIP has focused upon the development of an improved user-interface and a resource model which models the capacities and limitations of the package technologies. The new user interface is built as an RPC application to the VEM, the Octtools' graphical editor and is a significant improvement over the old user interface. We hope that this interface will allow others to evaluate SLIP and to give us feedback and access to examples.

The resource model provides a representation of packaging technologies which is simple enough to be used as a basis for optimization, yet sufficiently general to model most real-world technologies. This model relies upon user-defined callback functions to evaluate package resources, and so should enable SLIP to be easily configured to new technologies.

The user interface and the package resource model are near completion, although more work is expected as users adopt SLIP and more realistic packaging technologies are modeled. In the near future, we hope to model an industrial ASIC technology and to test SLIP by partitioning a set of examples into this technology.

Our long-term goals are to develop a timing model in SLIP and to develop delay-based partitioning methods which will minimize the effect of partitioning upon the system's performance.

1.17. Applying Synthesis Techniques to Aid Simulation (A.R. Newton)

Simulation is an important tool in many areas of circuit design. This project focuses on the simulation of a design specification, where the goal is to determine if the specification fits the need for which it was developed (as opposed to simulation whose goal is to determine if a design matches its specification).

In common with all simulators, those operating on specifications are generally slower than one would like. One technique that has been used to speed up all types of simulators is to produce program fragments to evaluate functions that need to be simulated, and compile these fragments into machine code for some target computer. Although such a "compiled-code" simulator gains some speed benefit, it generally simulates the specification as written, which was not optimized to make the best use of the characteristics of the target machine. In fact, since it is a specification, it is only optimized for clarity. This project seeks improvements in simulation

speed by treating simulation as a circuit synthesis problem, where the objective is to create a "circuit" that will simulate at high speed on the target machine, rather than to create a circuit that takes up a small area on a chip. Although these simulation goals are frequently similar, they are not necessarily so. If the target machine is a massively parallel single-instruction/multiple-data machine, for example, a significant advantage may be gained by mapping the design onto many instances of a single primitive function, so that all the processors can be doing the same thing all the time. On a conventional machine, it may be important to structure the "circuit" so that its width is limited, allowing the computer to hold the temporary variables that represent internal signals in its high-speed registers.

To date, a program has been written to create a compiled-code simulation of combinational logic blocks using three-value logic simulation. Fairly conventional logic optimization techniques have been applied, and seem generally to improve the simulation speed by a factor of 2-5 for the larger examples studied. Current work involves partitioning the initial design and optimizing the parts: this seems to reduce the simulation speed on the order of 20%, while decreasing the time spent in optimization by a factor varying from 1 to 2. It has the important benefit of allowing the optimization to complete on more examples without running out of memory. In the coming months, we plan to port the simulation to a massively parallel machine, and investigate more machine-specific optimizations.

1.18. A Generalized Approach to the Constrained Cubical Embedding Problem (A.R. Newton)

In this research, an efficient generalized approach to the constrained cubical embedding problem is proposed. Optimal cubical embedding is tightly related to various encoding and state assignment problems in high-level synthesis. The goal of cubical embedding aims at the embedding of symbolic values onto the vertices of a Boolean hypercube based on the satisfaction of constraints and some objective criterion. This is a known difficult combinatorial optimization problem whose complexity has been shown to be NP-complete. [40].

Previous constrained cubical embedding algorithms were formulated to solve specialized constraints. These constraints result from different symbolic minimization procedures and problem formulations. For example, in [41], a symbolic minimization procedure for two-level state assignment that considers the input fields in the state transition table was proposed. Two-level multiple valued minimization was used to obtain a reduced symbolic cover along with a set of embedding constraints. These constraints, called "face embedding constraints," are to be satisfied in the embedding process to obtain a compatible Boolean cover.

In [42], an extended two-level symbolic minimization procedure was proposed for the encoding of symbolic outputs. The corresponding output constraints, called "output dominance constraints," are generated in the minimization process. Algorithms have been proposed to solve these output constraints [43], but combining them with input constraints has been unsatisfactory.

While specialized algorithms are effective for their restricted domains, generalizing them to tackle a variety or combination of constraints is a formidable task since the nature of these constraints are quite different. Therefore, a robust generalized approach would be desirable for solving the different embedding problems. It should be easily extendible to problem formulations and cost objectives.

Probabilistic hill climbing has proved to be an effective approach to complex combinatorial problems in design automation [44]. Depending on the nature of the configuration space, probabilistic hill climbing techniques have been shown, in fact, to produce superior results to specialized algorithmic approaches on some combinatorial problems with comparable cpu time.

In this work, a new generalized framework for the constrained cubical embedding problem using probabilistic hill climbing techniques has been developed. Although probabilistic hill climbing in general may not be suitable to all combinatorial optimization problems, our results strongly demonstrate that it is extremely effective for this particular problem domain.

The approach has been implemented in a package called CUBIC. The generalized solver is separated from the constraints so that application specific constraints and objective criteria can be incorporated in a straightforward manner. We have obtained experimental results on a large set of design examples using our generalized approach. The results obtained are comparable or superior to those by specialized methods in similar cpu time.

1.19. Symbolic Encoding of High-Level Descriptions for Multi-Level Implementations (A.R. Newton)

In this research, the problem of symbolic encoding for automated synthesis of high-level descriptions is addressed. Well established logic synthesis procedures have been developed for implementations on two-level macros such as PLAs [49]. These techniques are aimed at minimizing the number of product terms in the final output. More recently, considerable attention has been devoted to the development of multi-level optimization procedures for multi-level implementations such as standard cells or CMOS complex gate technologies [45].

In standard logic synthesis systems, the input is a Boolean description in the form of either a binary truth table or a high-level specification using a Hardware Description Language (HDL). At Berkeley, the input specification to our logic synthesis system MIS can be a high-level description written in the BDS language. Since the description is a binary representation, the only data types allowed are bit-vectors of binary values. However, in high-level descriptions, the ability to represent the values of some signals at a higher level of abstraction would be greatly desirable. As an example, it would be desirable to represent the internal states of a finite state machine description with mnemonics (strings of characters) rather than forcing the user to specify the binary encodings. For a custom microprocessor, it may be desirable to represent the values of the instruction stream with mnemonics (eg. ADD, COMP, BRANCH, etc). But more importantly, the result of the logic optimization procedure is heavily dependent on the encodings of these symbolic values.

Therefore, we are developing a Hardware Description Language that allows the designer to specify as many different sets of symbolic values as required. Each of these sets, called a symbolic type, describes the admissible values allowed. Symbolic variables can then be defined using these symbolic types. A special symbolic type is the Boolean type which has the admissible values {0, 1} corresponding to binary logic descriptions. As part of a synthesis system, a symbolic encoding procedure has been developed that attempts to optimally encode symbolic values into binary bit-vectors. This optimization procedure has been implemented in a program called JEDI [50].

A special case of the symbolic encoding problem is the standard state assignment problem in which only the state variables are symbolic assuming one set of admissible values. The problem we address here is considerably more general. As such, our symbolic encoding procedure can effectively solve the state assignment problem as well. Optimal state assignment procedures have been developed for both two-level and multi-level implementations [47][46]. The objective criterion previously used for two-level optimization is the final area of a PLA implementation using the number of product-terms as an approximate indicator. For the multi-level case, a de facto objective criterion is to minimize the literal count.

For the general problem of symbolic encoding, an optimization procedure targeted for two-level implementations has been described in [48]. Our symbolic encoding procedure is targeted for multi-level implementations. As with multi-level state assignment, we use the final literal count after logic minimization as the optimization criterion.

1.20. Exploring Equivalent State Machines and State Assignment (A.R. Newton)

Finite state machines (FSM) can be specified in the form of a state transition graph (STG). The eventual implementation of the state machine is in the form of a sequential logic network consisting of combinational logic gates and synchronous registers. Given a finite state machine specification, there may be many possible implementations. The goal of synthesis is to find the one with the least cost.

Traditionally, this step is performed via an optimization process called state assignment. The goal of state assignment is to optimally assign binary codes to the internal states of a finite state machine such that the resulting synthesized logic is minimized. The assignment is restricted such that no two states are assigned the same binary combination. The problem of optimal state assignment has been a subject of extensive research. Numerous techniques have been developed to solve this problem. Most notably are contemporary methods that have a tight coupling to the underlying logic synthesis algorithms as to assure the optimality of the encoded results.

While effective, traditional state assignment techniques do not explore possible state assignments from other equivalent state machine specifications. Therefore, the solutions obtainable via standard state assignment may be suboptimal. That is, a better realization may have been possible from an equivalent machine with a different structure. In general, two machines can exhibit the same overall terminal behavior even though their respective structures are different (non-isomorphic).

Equivalent machines can be realized by merging or splitting states. We refer to such transformations as restructuring. State reduction techniques have been proposed for modifying the machine structure by identifying equivalent states and systematically eliminating them. The primary objective of state reduction is to obtain an equivalent machine with the minimal number of states. However, it is well known that a reduced machine does not necessarily lead to a better logic implementation and thus offers no guarantee as to the quality of the final solution.

In this work, the relationship between state machine restructuring and state assignment is explored. One approach is to combine these steps. Thus, rather than using the number of states to guide the state reduction process, we hope to make restructuring decisions implicitly via a more general state encoding problem.

REFERENCES

[1] T. Toyabe, H. Masuda, Y. Aoki, H. Shukuri, T. Hagiwara, "Three-Dimensional Device Simulation CADDETH with Highly Convergent Matrix Solution Algorithms", *IEEE Trans. on Electroon Devices*, Vol. ED-32, pp. 2038-2044, 1985

[2] R. Guerrieri, A. Sangiovanni-Vincentelli, "Three-Dimensional Capacitance Evaluation on a Connection Machine", *IEEE Trans. on CAD, Vol. 7, pp. 1125-1133, 1988*

[3] D. Hillis, *The Connection Machine*, MIT Press, Cambridge, Massachusetts, 1985

[4] D. Scharfetter, H. Gummel, *IEEE Trans. on Electr. Dev.*, Vol. 16, pp.64-77, 1969

[5] Kenneth S. Kundert and Alberto Sangiovanni-Vincentelli, "Simulation of Nonlinear Circuitts in the Frequency Domain", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-5, No.4, October 1986, pp. 521-535.

[6] Kenneth S. Kundert, Gregory B. Sorkin and Alberto Sangiovanni-Vincentelli, "Applying Harmonic Balancce to Almost-Periodic Circuits" *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-36, No. 2, February 1988, pp. 366--378.

[7] Kennth S. Kundert's Ph.D. Thesis, "Steady-State Methods for Simulating Analog Circuits, Chapter 8.

[8] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint Generation for Routing Analog Circuits " to appear in *Proc. Design Automation Conference*, June, 1990

[9] U. Choudhury and A. Sangiovanni-Vincentelli, "Use of Performance Sensitivities in Routing of Analog Circuits", *Proc. International Symposium* in May, 1990

[10] Sharad Malik, Ellen Sentovich, Robert K. Brayton and Alberto Sangiovanni-Vincentelli, "Retiming and Resynthesis: Optimizing Sequential Networks using Combinational Techniques", *Proceedings of the Hawaii Conference on System Sciences, 1990.*

[11] H. Touati and R. Brayton, "Retiming and Initialization of Sequential Circuits", in preparation.

[12] H. Touati, C. Moon, A. Wang and R. Brayton, "Performance-Oriented Technology Mapping", In *Proceedings of the Sixth MIT VLSI Conference, April 1990.*

[13] Richard Rudell, "Logic Synthesis for VLSI Design", Ph.D. Thesis, U.C. Berkeley, 1989. Memorandum No. UCB/ERL M89/49

[14] P. Ashar, S. Devadas and A. R. Newton. "Optimum and Heuristic Algorithms for Finite State Machine Decomposition and Partitioning," *Int'l Conference on Computer-Aided Design* November, 1989.

[15] J. Hartmanis and R. E. Stearns, "Algebraic Structure Theory of Sequential Machines," Prentice-Hall, Englewood Cliffs, N. J., 1966.

[16] F. C. Hennie, "Finite-State Models for Logical Machines," Wiley, New York, 1968.

[17] S. Devadas and A. R. Newton, "Decomposition and Factorization of Sequential Finite State Machines,", IEEE Transactions on CAD, *IEEE Transactions on CAD*, to appear November, 1989.

[18] S. Malik, E. Sentovich, R. Brayton and A. Sangiovanni-Vincentelli, "Retiming and Resynthesis: Optimizing Sequential Circuits Using Combinational Techniques,", *Proc. of 1989 MCNC Logic Synthesis Workshop, May, 1989.*

[19] P. Ashar, S. Devadas and A. R. Newton. "A Unified Approach to the Decomposition and Re-decomposition of Sequential Machines," *In Proceedings of the Design Automation Conference, June 1990.*

[20] F. C. Hennie, "Finite-State Models for Logical Machines," New York, 1968.

[21] S. Devadas and A. R. Newton, "Decomposition and Factorization of Sequential Finite State Machines," *IEEE Transactions on CAD, to appear November 19889.*

[22] S. Malik, E. Sentovich, R. Brayton and A. Sangiovanni-Vincentelli, "Retiming and Resynthesis: Optimizing Sequential Circuits Using Combinational Techniques," *Proc. of 1989 MCNC Logic Synthesis Workshop.* May 1989.

[23] J. Hartmanis and R. E. Stearns, "Algebraic Structure Theory of Sequential Machines," Prentice-Hall, Englewood Cliffs, N. J., 1966.

[24] S. Devadas, H-K. T. Ma, A. R. Newton and A. Sangiovanni-Vincentelli, "Irredundant Sequential Machines Via Optimal Logic Synthesis," *IEEE Transactions on CAD, January 1990.*

[25] P. Ashar, S. Devadas and A. R. Newton. "Testability Driven Synthesis of Interacting Sequential Machines, " Submitted to *Int. Conf. on Computer Design* February 1990.

[26] P. Bitar, M. Despain, "Multiprocessor Cache Synchronization: Issues, Innovations, Evolution",

Proceedings of the International Symposium on Computer Architecture pp 424-433, June, 1986

[27] P. Sweazey, A.J. Smith, "A Class of Compatible Cache Consistency Protocols and Their Support by the IEEE Futurebus", *Proceedings of the International Symposium on Computer Architecture*, pp 414-423,1986

[28] J. Archibald, J-L. Baer, "An Economical Solution to the Cache Coherence Problem", *Proceedings of the International Symposium on Computer Architecture*, pp 355-362, 1984.

[29] A, Gottlieb et. al., "The NYU Ultracomputer--Designing an MIMD Shared Memory Parallel Computer, *IEEE Transactions on Computers*, Vol C-32, No. 2, pp 175-189, February, 1983

[30] G.F. Pfister et. al, "The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture, *Procceedings of the International Symposium on Computer Architecture*, June, 1985

[31] D. Gajski et. al, "Cedar--A large Scale Multiprocessor, *Proceedings of the International Conference on Parallel Processing*, pp 514-529, August, 1983

[32] A.W. Wilson Jr., "Hierarchical Cache/Bus Architecture for Shared Memory Multiprocessors, *Proceedings of the International Symposium on Computer Architecture*, pp 244-252, June, 1987

[33] J.R. Goodman, P.J. Woest. "The Wisconsin Multicube: A New Large-Scale Cache-Coherent Multiprocessor, *Proceedings of the International Symposium on Computer Architecture* pp 422-433, May, 1988

[34] M. Carlton, "Efficient Cache Coherency for Multiple-bus Multiprocessor Architectures *Dissertation Proposal*, November, 1988

[35] B. O'Krafka, "An Empirical Study of Three Hardware Cache Consistency Schemes for Large Shared Memory Multiprocessors, *M.S. Report*, U.C. Berkeley, March 1989.

[36] M. Censier, P. Feautrier, "A New Solution to Coherence Problems in Multicache Systems, *IEEE Transactions on Computers*, Vol. C-27, No. 12, pp 1112-11188, December, 1978

[37] B. O'Krafka, A.R. Newton, "An Empirical Study of Two Memory-Efficient Directory Methods", To appear in *Proceedings of the International Symposium on Computer Architecture, May 1989.*

[38] M. Beardslee, C. Kring, R. Murgai, H. Savoj, R. K. Brayton and A. R. Newton, "SLIP: A Software Environment for System Level Interactive Partitioning", To appear *Proc. 1989 IEEE International Conference on Computer-Aided Design*

[39] D. Harrison, P. Moore, R. Spickelmier and A. R. Newton, "Data Management and Graphics Editing in the Berkeley Design Environment", *Proc. 1986 IEEE International Conference on Computer-Aided Design*, pp 20-24, Nov. 1986

[40] B. Lin and A.R. Newton, "A Generalized Approach to the Constrained Cubical Embedding Problem", International Conference on Computer Design, October 1989.

[41] G. DeMicheli, R. Brayton, and A. Sangiovanni-Vincentelli, "Optimal State Assignment of Finite State Machines", *IEEE Transactions on CAD*, pp 269-285 July 1985.

[42] G. DeMicheli, "Symbolic Design of Combinational and Sequential Logic Circuits Implemented by Two-Level Macros", *IEEE Transactions on CAD*. pp. 597-616, October 1986.

[43] T. Villa and A. Sangiovanni-Vincentelli, "Nova: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementations", *Proceedings of the Design Automation Conference, June 1989.*

[44] S. Kirkpatrick, C. Gelatt, Jr., and M. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, pp. 671-680, May 1983.

[45] R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli and A. Wang, "MIS: A Multiple Level Logic Optimization System", *IEEE Transactions on CAD* November 1987.

[46] S. Devadas, H.T. Ma, A.R. Newton and A. Sangiovanni-Vincentelli, "MUSTANG: State Assignment of Finite State Machines for Optimal Multi-Level Logic Implementations", *Proc. of Int'l Conference on Computer-Aided Design*, Santa Clara, November 1987.

[47] G. DeMicheli, R.K. Brayton and A. Sangiovanni-Vincentelli, "Optimal State Assignment of Finite State Machines", *IEEE Transactions on CAD* Volume 1, Pages 269-285, July 1985.

[48] G. DeMicheli, "Symbolic Design of Combinational and Sequential Logic Circuits implemented by Two-level Macros", *IEEE Transactions on CAD*, Volume 1, Pages 597-616, October 1986.

[49] R. Rudell and A. Sangiovanni-Vincentelli, "Multiple Valued Minimization for PLA Optimization", IEEE Transactions on CAD, *IEEE Transactions on CAD*, Volume 1, Pages 727-757, September 1987.

[50] B. Lin and A.R. Newton, "Synthesis of Multiple Level Logic from Symbolic High-Level Description Languages", *VLSI Conference, August 1989.*

2. ARCHITECTURES AND APPLICATIONS

2.1. VLSI ASIC and System for Hard Real-Time Tracking (R.W. Brodersen)

Accurate detection and tracking of moving objects at video rates is an important problem in the vision control of robots and in other machine-vision tasks. In real-world applications, variable lighting conditions, video noise, properties of the objects, and the background of the scene further complicate this task. In particular, vision-controlled robots require highly accurate real-time information about their work scene.

Standard low-level (early-vision) image-processing algorithms typically require highly controlled lighting and noise-free images to achieve real-time or near real-time recognition of objects in the scene. Transforming images into the Radon space allows these and other constraints on the images to be relaxed. The Radon transform provides robust recognition of lines and line segments in images in the presense of noise and suboptimal lighting. Recognition and tracking of objects then proceeds in the Radon space.

Objects are trained into the system by performing detailed analysis of their representation in the Radon space. Depending on the accuracy required for the particular application, the objects can then be recognized by a variety of algorithms in the Radon domain. Once the system locks on an object, tracking the object involves incremental analyses of slices through the Radon domain.

A new, highly integrated 9U VME-based board has been built that implements the Radon transform in real time. Each board has four complete custom AT&T DSP32C microprocessor cores, each of which serve as a local host. Each of these DSPs provide the necessary modularity and programmability to support eight custom Radon-transform application-specific ICs (ASICs), for a total of 32 ASICs on a board. Parallel and pipelined processing occurs at both the chip and the board level. Multiple Radon-transform boards are installed in a single 21-slot VME card cage, for a total processing power capability of one-quarter trillion operations per second. The card cage has a 68020-based single-board computer and an Ethernet communications card to provide support as a global host to all of the slave processors and local-area network support to nearby workstations. The card cage runs a real-time operating system that resembles UNIX to facilitate low-latency transfers of recognition primitives over the VME bus. Communications and control of the image-processing cards occurs via remote workstation through RPC and X-windows.

The VLSI ASIC that executes the highly computational- and I/O-intensive Radon transform algorithm has been fabricated in a 1.6-μm process through MOSIS. It operates at the required 10-MHz video rate, and consists of over 100,000 transistors. Its internal architecture is flexible; can be programmed externally through the DSP32C to perform both forward and reverse Radon transforms and grey-level histograms and to act as a general-purpose RAM. Built-in-line-length counters facilitate line-length normalization. Hardware support is also provided to allow the region of interest over which the Radon transform is taken is to be detrermined externally. The ASIC supports both interlace and noninterlace image formats.

Specific applications of the image-processing board include accurate positioning of a four-degree-of-freedom manipulator for laser-reconfigurable integrated circuits. A video camera attached to a microscope/probe station allows the system to recognize objects in the circuit and to correct the position of the IC. Translation and orientation are determined at the video frame rate. The system can teach and recognize a number of different styles of objects; thus the image-processing algorithms can effectively increase the accuracy of the manipulator to that of its resolution.

Other applications of the tracking system include a six-degree-of-freedom robot with custom control hardware that is integrated with the real-time image processing hardware.

2.2. Real-Time, Flexible Image-Processing Printed Circuit Board (R.W. Brodersen)

This project integrates many real-time image processing modules into a complete, compact system. The system incorporates (1) a custom VLSI low-level image-processing chip set, as designed by Ruetz [1], (2) a custom VLSI histogram processor and equalization chip set, as designed by Richards [2], and (3) support for an external image processor board.

Any or all of the three image processors can be used, in any order, with commercial A/D-D/A and frame buffer-boards. This system can be reconfigured with the aid of two custom VLSI video crossbar switches, designed and fabricated in a 2.0-μm CMOS process. These two application-specific IC designs, fabricated in 132 and 68-pin packages, are fabricated and are fully functional at 10MHz video retes. Four of the 132-pin version and one of the 68-pin version provide 24-bit (color) and 8-bit multiplexing of video buses as required by the chosen image-processing algorithm. Internal pipeline registers ensure 10-MHz throughput, and an external processor can write to the internal configuration registers.

The entire-image processing system has been fabricated on a 9U VME board and is fully functional at video rates. It contains 16 custom ASICs and 2 programmable logic devices (PLDs) for interface logic. A slave VME bus interface provides control of all the image processors and video multiplexer configurations. The board resides in a 21-slot stand-alone card-cage along with a 68020-based CPU card, an Ethernet card, and a custom robot controller card [3]. The board is connected by Ethernet to a Sun workstation and is controlled through a set of X-window front-end tools. Window-based software has been written to reconfigure the processors in any arbitrary fashion.

2.3. Real Time Image Data Compression (R.W. Brodersen)

This research investigates and extends existing algorithms and architectures for image data compression. The study is geared towards full-motion video transmission, which requires compression from an initial rate of 62.9 Mbits/s (for images defined over a 512 × 512 lattice, assuming 8 bits per pixel and 30 frames per second) to a reduced rate of 1.5 Mbits per second (i.e., a 40:1 compression). Color images require higher compression rates. For this application, the compression algorithms are constrained by the need for real time performance and low power dissipation.

The two traditional classes of compression techniques (transform coding and spatial domain coding) are being simulated with special emphasis on interframe hybrid transform coding. In this coding scheme, a two-dimensional transform, typically the discrete cosine transform (DCT) is performed on sub-blocks of the image, and the error signal from recursive estimation of previously transmitted transform coefficients is transmitted. This method exploits the spatial redundancy within the image to achieve high compression rates. Other techniques that improve compression, such as motion detection and estimaton , are also to be included. A major concern of this research is to adapt appropriate image-compression algorithms for efficient VLSI implementation. Low power design techniques and technology factors such as scaling will play a major part in the development of a low power and compact compression system.

A prototype real time image compression/decompression system is currently being designed. The system implements a generic architecture capable of simulating various image compression algorithms in real time including intraframe coding, interframe coding, and motion compensation. Special emphasis will be placed on the new CCITT standard for full motion image data compression at multiples of 64 kilobits per second. The board is being desined with commercial custom compression chip set including a DCT (and inverse DCT) processor, a Motion Estimation Processor , and a Quantization Processor. It will also contain local video frame buffers, and signal processors to perform various miscellaneous tasks such as image data indexing, and variable length coding. Software control will be used to reconfigure the board for different image compression algorithms.

2.4. Programmable IC Digital Signal Processor with Self-Timed Internal Processing Elements (R.W. Brodersen)

While digital signal processing algorithms for many applications have been under study for decades, it is only recently that a variety of general purpose digital signal processor (DSP) chips have become available to execute these algorithms. Many products, especially in the telecommunications area have begun to employ DSPs in their design. It is thought that the continued scaling of the IC process, which yields higher density and faster transistors, should allow even more sophisticated algorithm implementation. However, limitations in the way scaling can be performed along with continued growth in chip areas may interfere with the ability to take advantage of the fast devices. Namely, the time taken for signals to traverse a chip through the interconnection layers is becoming significant with respect to the clock periods of processors. Using

today's synchronous "clocked" design techniques will probably not be adequate in the future because of the difficulty in retaining global synchronization as clock rates exceed 100MHz.

The use of self-timed circuits (circuits that generate completion information) to construct fully asynchronous processors is one possible solution to the synchronization problem mentioned above. This idea forces timing signals to be generated locally, bringing the synchronization problem back down to a manageable size. The circuitry also monitors its own timing signals so that it can absorb variations in time used for logic signal evaluation and traversal through interconnect and still function correctly. In this way, the clock distribution problem can be alleviated and the design of a digital signal processor will truly be scalable with the technology advancements. Also, since a clocked system inherently cannot take advantage of the differences in delay associated with different elements of a signal processor (such as ALU, shifter, multiplier), the self-timed approach may be able to take advantage of these differences to obtain a faster cycle time.

In an asynchronous system, each block generates a completion signal to indicate that it has performed its task. Interface circuits (also known as handshaking circuits) make use of these completion signals to control the proper transfer of data between stages. The circuit designer need only be concerned with the operation of each individual block or macrocell as is typically done is lower speed clocked designs where global synchronization is not an issue. Also, by separating the interface circuits from the computation circuits, system timing becomes a matter of designing only the proper collection of interface circuits that realize the desired system operation.

Self-timed datapath macrocells were developed, fabricated and tested. Each of these implements a datapath function such as shifter, multiplier, and ALU and provides completion information. Combining the datapath cells with RAM and ROM cells, a fully asynchronous ROM programmable digital signal processor was designed and fabricated. This is the first implementation known of a fully asynchronous DSP that contains feedback and that is programmable. In the DSP, a data stationary architecture is used, where the control signals for each stage of the datapath pipeline move along with the data flowing through the datapath. This fits the self-timed paradigm well since the data transfers themselves occurr at times not synchronized with some global clock. The I/O ports on the DSP are also self-timed so that the processor will automatically wait for external devices to be ready for transfers.

Three versions of the self-timed DSP were fabricated. Each was programmed to perform a simple signal processing task. The first chip implements a 16-tap FIR lowpass filter, the second chip implements an 8-pole IIR bandpass filter, and the third chip contains a test program to determine that the functionality of all of the different types of instructions is correct. The instruction cycle time depends both on the type of instruction and the data in the datapath. The basic "fast" instruction is a shift/add operation. A hardware multiplier is available too, but since it performs multiplication in an iterative fashion, the cycle time when it is used is longer. Other differences in cycle time are attributed to the carry-propogation time in the ALU adder circuit, which is data-dependent. The self-timed DSP was fabricated in 2um N-well MOSIS process and a microphotograph is shown in Figure 1. Measured cycle times were roughly 75nsec for the fast instruction and 325nsec for a multiplication. Further gains over synchronous designs are expected in a more highly scaled technology.

The asynchronous DSP has been described in two recent publications: "A Fully Asynchronous Digital Signal Processor using Self-Timed Circuits" was presented at the 1990 *IEEE International Solid State Circuit Conference* in San Francisco, CA. "Self-Timed Integrated Circuits

for Digital Signal Processing" was published as a Memo of the Electronics Research Laboratory at UC Berkeley (Memorandum No. UCB/ERL M89/128). Another journal article describing the processor is currently being prepared and it will be submitted to the *IEEE Journal on Solid State Circuits*.

2.5. TADS: A Test Application and Development System (R.W. Brodersen)

As the complexity of integrated circuits increases and surface-mount interconnection technology evolves, significant new problems arise in testing of board-level systems. The IEEE Joint Test Action Group has proposed a design-for-testability standard which alleviates these problems. They have proposed a boundary-scan architecture which is comprised of a boundary-scan-register (BSR) and a test access port (TAP) that allows access both to the BSR and to other embedded test circuitry. These BSRs allow control and observation of any node in the system, improving its testability.

Incorporating this design-for-testability standard into our board-level designs requires a system that does it automatically. The system should also provide software that supervises and administers a test and configures the testing hardware. TADS, which is a TEST APPLICATION and DEVELOPMENT SYSTEM, is currently being developed to accomplish these tasks.

The current research has two objectives:

(1) The first objective is to design a test controller module and a library of test support cells. The test controller module interfaces with other modules via TAP bus and executes the tests.

(2) The second objective is to develop software that automatically generates test patterns, automatically synthesizes the test hardware, and analyzes the test results.

2.6. Design Tools for Rapid Design of Oversampling A/D Converters (R.W. Brodersen)

The goal of this project is to provide tools that will aid designers in exploring tradeoffs in the design of these circuits. Current methods of design rely on extensive time domain simulations and expertise in the areas of switched capacitor analog circuit design and digital filter design.

An evaluation was made of the current Berkeley tools for the synthesis of basic analog building blocks. These tools are lacking in generality and are not well tested. Since oversampling A/D converters require small amounts of analog circuitry that can be reused in many designs, it makes more sense to design leafcells which can be characterized through testing rather than trying to automate the analog design process for each new design. To this end, a second order delta sigma modulator has been fabricated and will be tested and characterized in the coming months.

Currently, a collection of C language programs exists for simulating the modulators, but the previous hardware platform used for circuit testing and data acquisition has become out ated. A new set of testing boards is being designed. The goal is to provide a means of collecting data from actual circuits and uploading the data through a VME interface. The simulation and anlysis programs will then be used to characterize circuit behavior.

Oversampling A/D converters require digital filters for eliminating out of band noise prior to resampling at a lower rate. These filters tend to be costly in area. An area efficient architecture has been identified for the first digital filter that processes the 1 bit output of the modulator. Efforts are being made to write a program that can map a finite impluse response onto this architecture automatically, providing sdl files as output.

2.7. Active-Word Processor for a Real-Time, Large-Vocabulary, Continuous Speech Recognition System (R.W. Brodersen)

The active-word processor controls the active-word and active-list memories. Data comes from two sources, the Viterbi processor and the grammar subsystem. The former requests that a currently-active word be added to the active-word memory if this word is likely to still be active during the next frame; the latter requests additions to the active-word memory if a highly probable word, having just ended, generates one or more successors. These requests may come at any time so the active-word processor has to arbitrate between requests in order to access each memory at most once per clock cycle. The processor must therefore stall the Viterbi processor and/or refuse data from the grammar subsystem when necessary.

The processor uses data stations to store requests. When a request arrives, it is loaded into the first available station. Once in the station, it is processed automatically. Processing requires 6 clock cycles, after which the station is available for the next request. Each request requires two accesses to the active-word memory and two accesses to the active-list memory. Therefore, 3 data stations are required. If a newly-loaded request conflicts with another request that is being processed in another station, the new request is ignored until the other request is completed.

There are several datapaths in the processor. They are the grammarnode probability datapath (16 bits), wordarc datapath (20 bits), tag datapath (20 bits), topology datapath (16 bits), state probability datapath (18 bits), phoneme datapath (5 bits) and flag datapath (2 bits). There is also a counter and a control unit containing a PLA-based finite state machine.

The active-word processor is physically implemented in two custom chips. This is necessary because the pincount of the processor is too high to fit into one package. The larger chip contains the state machine that controls the operation of the processor and the datapaths for the flags, wordarc and grammarnode probability. The smaller chip is a slave to the larger chip and contains all the other datapaths.

Some additional circuitry is included in the chips to facilitate testing. To aid in chip testing, all registers are scanpath registers. For board testing, the chips interface between the VME bus and the two memories so that these memories may be written and read by the VME host.

The chips are being designed using the LagerIV silicon assembly system and will be simulated using IRSIM.

2.8. Interface Board for the Robot Control System (R.W. Brodersen)

This project involves the design of an interface board for the robot system. The board interfaces the various relays, motor current sensors and H-bridges driving the robot motors with the VME board running the control algorithms. The main motivations behind the design of this board are to isolate the high-speed digital electronics on the VME board from the noisy high-power electro-mechanical devices in the robot and to improve the mechanical characteristics of the system. This is being done to alleviate the electro-magnetic interference and ground-bounce problems experienced in our current implementation where the interface electronics is on the VME board itself. A second motivation for the board is to provide current feedback to the H-bridges allowing control of the motor torques, as opposed to the voltages, which is desirable for most sophisticated robot control algorithms.

The board consists of analog filters, comparators and pulse-width modulation circuits to provide current feed-back based control of the H-bridges for each of the six joints of the robot arm. In addition, six analog-to-digital converters and some digital circuits allow digital generation of the pulse-width modulated signals on the VME board itself and torque control through digital current feedback.

Optical isolation, shielding and multiple ground and power planes are being used to isolate the VME board from the robot and to reduce ground bounce and EMI on the interface board itself. The schematic design is complete and the board place-and-routing is expected to be done in a few weeks.

Simultaneous to the current design we are exploring the use of time-multiplexed optical fiber based communication between the VME board and the interface board. This will result in much increased EMI resistance, mechanical robustness and isolation. Right now we are looking into possible techniques and the implications on the system architecture and partitioning.

2.9. Design of Real-Time Systems with Application to Robotics (R.W. Brodersen)

The goal of this project is to develop a CAD framework for the design of dedicated real-time systems starting from a high level structural description. In particular, we are interested in real-time feedback systems that interact with the outside world through a variety of asynchronous events. A real-time multi-sensory robot control system is being used as the driver application.

To study first hand the issues involved in such systems, a manual design of a first-generation robot control system was done the last year. A DSP32C-based custom VME slave board forms the heart of the system and is fully operational. Our experience indicated that simulation and the integration (communication and synchronization) of the various software and hardware modules residing on different custom boards or general purpose CPUs forms the most difficult part of the design phase.

Making use of our experience we are currently exploring a suitable CAD framework for the design of such systems. Our experience suggests that a good way of describing such complicated systems is to view them as a static network of concurrent processes interacting with each other through events and messages. The processes can be implemented as software running on general

purpose CPUs or as dedicated VLSIs. More than one process may be mapped on to a single general purpose CPU or custom VLSI. Such a description can be viewed as a type of block diagram. Currently we are looking into representation in OCT and simulation of systems described in such a fashion.

Next we plan to work on mapping such a description into a library of hardware and software modules and automatically generating the 'glue' software and hardware to handle the communication and synchronization. Our approach is based on defining a generic architecture for multi-board systems, which consists of multiple slave boards on a VME backplane that have standardized hardware and software interfaces to a VME master CPU. The slave boards are custom boards implementing complete subsystems or are off-the-shelf CPU cards. Each custom board is in turn based on a core architecture consisting of a board-controller CPU and multiple slave modules. The slave modules come from an application-domain-specific library and can be software for a general-purpose DSP or microprocessor or behavioral or structural description of an ASIC.

We intend to use the CAD framework to describe, simulate and design our next-generation robot-control board which will have more compute power than the current board and ability to handle force sensor data.

2.10. Backtrace Memory Processor for a Real-Time, Large-Vocabulary, Continuous-Speech Recognition System" (R.W. Brodersen)

A crucial part of the proposed real-time continuous-speech recognition system [9] is the backtracing algorithm to recover the state sequence through the grammar model after a whole sentence has been spoken. Since this sequence is not known until the last frame of a sentence has been processed, all the possible word sequences have to be stored in a memory (backtrace memory). Each entry in this memory contains the identification of a particular grammarnode along with an address that points to the predecessor of this grammarnode. Since there is no end-of-word detection, each grammarnode needs an entry in this memory for every timeframe. Thus, the stored predecessor of a grammarnode is in most cases the grammarnode itself.

To avoid storing too much backtrace information, the backtrace memory processor only stores a grammarnode in the backtrace memory if the probability associated with this grammarnode is higher than a threshold probability. The processor also computes this pruning threshold based on a running maximum probability. The running maximum can be preset at the start of each frame to avoid initialization effects.

The processor is a pure slave processor to the Viterbi processor. It is not driven by the system clock; a strobe generated by the Viterbi processor starts a pruning and threshold-updating operation after a word has been processed.

Because of this asynchronous behavior and the low computational rate (once for each grammarnode), a standard-cell design methodology was used. The backtrace memory processor, which was fabricated in 2-μm CMOS technology, has a die size of 3044 μm × 3574 μm, 7092 transistors, and 94 signal pins. It is packaged in a 108-pin pin-grid array. All layout was generated using the LagerIV silicon assembly system.

2.11. ASIC's for Numerical Computations (R.W. Brodersen)

The traditional problem domain for digital signal processors (DSPs) has been operations such as filtering, equalization, and spectrum analysis. This project attempts to extend the boundaries of DSP applications to include more numerical computation problems.

As a development vehicle we are considering the problem of finding all solutions of a system of polynomial equations in n variables. This problem is irregular and complex compared to traditional digital signal processing, but also computationally intensive.

Our current work focuses on continuation methods [4]. The main problem is the need to solve large numbers of n by n linear equations. Whether it will be feasible to perform these computations in fixed-point arithmetic is not yet clear. For efficiency, it may also be necessary to develop a dedicated architecture for the problem.

2.12. ASIC's for Inverse Kinematics (R.W. Brodersen)

We are using the LagerIV silicon assembly system to design an application-specific integrated circuit (ASIC) for high-speed inverse kinematic computation for elbow-type robots such as the Puma 560 and the Panasonic NM-6740.

The goal is to produce a circuit that provides solutions at the sample rate (which may be up to 1 or 2 kHz) of the robot control loop. Such a circuit would make it possible to plan detailed trajectories in Cartesian space and feed the corresponding angle data directly into the control loop.

The algorithm is more complex and less structured than those commonly used in digital signal processing. In attempting to design an efficient and accurate fixed-point version of the algorithm, we have had to develop efficient implementations of elementary functions (atan2, sin, cos, and sqrt) for the LagerIV processor architecture. We have also identified and designed improvements to the processor architecture for the inverse kinematics task.

This project has been the first to use the RL high-level programming language to program a LagerIV processor. The RL compiler [5] translates RL directly into symbolic microcode. The retargeting ability of the compiler, which allows experimentation with several variants of the target architecture, has proved extremely useful for evaluating enhancements to the architecture.

New developments: The algorithmic problems have been solved, and our new facility for automatic assembly of THOR simulation models has been used to run complete simulations of the RL program. The results have been verified and compared to higher level simulations. The latest design improvements (see below) have been incorporated, and the final chip core has been successfully simulated with both THOR and IRSIM. We are currently working on fitting the core with pads. After a final from-the-pads simulation, the chip will be sent to fabrication.

2.13. A Programmable DSP for LagerIV (R.W. Brodersen)

The Kappa processor architecture was developed by Khalid Azim [6] for the LagerIII silicon assembler. We have ported the design to LagerIV and enhanced the architecture by including a newly designed logarithmic shifter and an array multiplier.

It is possible to generate different versions of the processor by selecting parameter values such as wordlengths, memory size, and microcode.

This process is automatic, driven by parameter values. It is also possible to customize such characteristics as the number of registers, multiplier type (array or shift/add), shifter type (variable shift from register vs fixed shift from instruction), and interconnections (buses). These changes require some manual work by the user.

Kappa can be programmed in the RL high-level programming language. [7] special-purpose architecture of Kappa. We have interfaced the standard version of Kappa to the RL microcode compiler [13] by providing a new Microcode assembler (Mass).

Our effort to automatically assemble THOR simulation models of any given instance of the Kappa processor is now sucessfully completed. This has made it possible to simulate the execution of complex RL programs directly on a logic level model of the processor, and at a reasonable speed (10 processor cycles per cpu second). This is more than an order of magnitude faster than layout extraction-based IRSIM simulations.

The access to speedy THOR simulations has prompted a more thorough testing of the architecture, and some fatal bugs in the original processor design have been corrected.

The internal control of Kappa has been redesigned to achieve higher layout density. This involved changes in the controller architecture, as well as design of new leafcells. The control unit is now 35% smaller than before and close to optimal for our largest test example.

The latest revision of the design has been brought through entire process from THOR simulation to layout, extraction and IRSIM simulation. The functionality of the new design has been verified.

2.14. Board-Level System Interfacing (R.W. Brodersen)

At the board level, where chips are the building blocks and are interfaced with extra logic, the development of board design techniques lags behind the development for VLSI chip design. To reduce board design complexity and to ensure functionality and required performance, this research aims to automate the design of interface logic and implementation. The chip components may use asynchronous or synchronous communication protocols like handshaking to transfer binary data between them. Given the protocols and the system architecture, i.e. communication topology of of who talks to whom, the goal of the interface tool is then to generate circuits that properly control protocol event sequencing and ensure that timing requirements are met during communications. This interface synthesis system provides high-level user input specification, a library database of components, and the actual synthesis tools.

The input from the user is a block diagram of the system architecture. The block diagram is made up of blocks which represent components or modules of components, and interconnection nodes and directed edges which represent interfaces and the logical communication channel that connects the module ports, respectively. The interface synthesis program can look up information necessary about the component's I/O timing behavior, as its protocol defines, which resides in a component library. The block diagram is represented as a netlist in the *.sdl* format.

The underlying library consists of components and information about their I/O timing behavior. I/O timing behavior normally contains many details about the sequencing of I/O events and timing relationships between events. These details are tedious for the user to enter for each design, and the library is provided as a convenience to the user and to maintain integrity of the timing behavior information. The timing behavior is represented as a signal-transition-graph[1] which has event nodes and precedence edges. Timing constraints between events are attached as weights to the graph edges. Currently, this research is exploring representation with the operation/event-graph model[8], which allows complex control over event sequencing. The graphs are stored in the database as a netlist of nodes and edges represented in a textual format, which is currently being developed.

The interface synthesis tool reads from the library the timing behavior of the appropriate components. Then given the communication topology from the block diagram, it can generate the interface logic. The first step is merging the event-graphs of individual component ports that communicate into one event-graph that represents the I/O timing behavior of the interface. The algorithm that merges the graphs will consider different ways of merging the graphs to get different performance and cost (of the interface circuit) results. In the next step, the synthesis software will include graph check programs which detect consistency errors or enforce a particular graph property to hold prior to actual synthesis. The goal here is to guarantee correctness of the logic implementation from the graph specification[1]. Thereupon, a functionally-correct merged graph specification can be transformed into a state graph from which logic synthesis can be performed using developed techniques[9].

2.15. Wideband Digital Portable Communications (R.W. Brodersen)

The purpose of this research is to investigate and develop techniques for digital wireless communications capable of handling extremely high data rates, yet in a small, portable device. Clearly, the desire for portability presents the challenge of low power consumption within the system, while the desired data rates (~10 Mbit/sec) present the conflicting goals of wide bandwidths and high data-processing rates. We envision such a device to serve as a terminal of a "micro-cellular" system, similar in spirit to today's cellular phone network, but with a fully digital implementation, capable of handling information transfers well beyond simple voice transmission, including full-motion video. The microcell concept allows for these bandwidth requirements, allowing high frequency reuse factors and correspondingly high spectrum efficiency, as well as keeping short transmission distances between mobile and base units.

We are now beginning to examine channel and data encoding, modulation, and detection schemes which are optimal within these constraints, through extensive use of computer-aided

modeling. Furthermore, channel noise, low-antenna multipath propagation, and fading issues in a microcell transmission environment must be considered. Currently, any such modelling must be done using field-measured fading/distortion data; we are developing a computer-aided simulation program which takes "physical descriptions" of the terrain (eg., placement of buildings, trees, etc) and provides reasonably accurate data concerning the transmission channel.

As a first iteration in developing such a communications system, we are constructing a 10 Mbit/sec wireless data link, which would be compatible with current Ethernet standards. The most promising candidate thus far is a time-division multiple access system, using 16- or 32- quadrature amplitude modulation with Viterbi error coding.

2.16. Oct2PCB Placement and Routing within LagerIV (R.W. Brodersen)

Until recently, the LagerIV assembly system addressed only the design of custom integrated circuits. However, most real-time systems, including those we are developing, incorporate commodity and programmable components, as well as custom devices on the same board. We have, therefore, extended the toolset of the LagerIV system to handle the design of printed circuit boards (PCBs). We have developed interfaces between the Oct database environment and a set of commercial PCB tools, such as those offered by Hewlett-Packard and Racal-REDAC, which allow us to describe the composition of the board using the sdl structural language exactly as we would describe the composition of a custom integrated circuit. This description, which might include some constraints on the board topology, is then passed to the commercial tools for placement and routing.

Current efforts focus on enhancing the modularity and features of the PCB description and adding tools for board extraction, modeling (including connectors and backplanes), and electrical simulation.

2.17. Techniques for Very Fast System Prototyping (J.M. Rabaey, R.W. Brodersen)

Prototyping a system is often the only way to test its correctness and optimize its behavior with respect to the initial specifications. System prototyping has traditionally been an extremely tedious operation, using commodity components such as microprocessors and memories combined with a lot of glue logic implemented in TTL.

Recent advances in technology have yielded several components and techniques that could reduce the prototyping time dramatically. One is the sea-of-gates approach, which allows the integration of systems of considerable complexity in a moderate turnaround time. Another approach involves the use of Programmable Logic Devices (PLDs), whose complexity has recently risen to 9000 gates. This approach offers great flexibility and instantaneous turnaround.

In this project, we have analyzed the use of PLDs for prototyping signal-processing applications such as speech recognition and image processing. The results of those experiments have allowed us to classify existing PLD devices according to their complexity, flexibility, and

architectural features. This information will help us to develop tools for automatic partitioning and architecture selection.

We have implemented an interface between the Lager/Oct environment and commercially available PLD mapping tools. We have also developed a method for mapping logic equations into PLD primitives using MIS-II and Espresso. To map large designs efficiently into a set of PLD's, we have developed two partitioning methods. One method uses a fast clustering algorithm. Another method uses a linear programming approach to find an optimal solution. With this interface, it will be possible to generate PLDs starting from our standard specification environment (sdl VEM) and to compare different implementation techniques (such as custom-cell, gate-array, or PLD) relatively quickly. We are currently targeting PLD's manufactured by Actel, Altera, and Xilinx.

We have used this interface to map designs used on a 32 megabyte memory board of our HMM speech recognition project. These designs include a memory controller, VME interface logic, address generation logic, and a saturating adder.

2.18. Fast Prototyping of Video and Speech Systems (J.M. Rabaey)

The implementation and realization of video and speech systems is always cumbersome, requiring extensive time and resources. The high performance constraints force the designer to use either a bulky TTL board or custom-designed chips, both of which are expensive.

This project attempts to speed up prototyping by defining a library of macrocomponents at a high enough level to simplify the board design but still flexible enough for a wide variety of high-performance signal-processing applications. To define the contents of the library and the desired programmability and functionality, we are currently examining a set of image- and video-processing algorithms as well as some problems in speech recognition. These include recursive filters, matrix conversion from RGB to luminance and chrominance for video, linear and non-linear image filtering using linear convolution and sorting algorithms respectively, dynamic time warping and Hidden Markov based algorithms for speech recognition, and others.

We have observed that most of these systems are implemented as a set of concurrently operating, bitsliced and pipelined-processors. The connection and communication patterns, the controller structure, the datapath composition, and the memory organization of the processors, however, depends heavily upon the application. For each entity, we are trying to define a restricted set of programmable components that covers most of the architectural alternatives. Four classes of devices are necessary: datapath blocks, controllers, memory (including delay lines), and interprocessor communication units.

Fairly efficient solutions are available for control structures (using programmable logic devices) and memory structures. However, no high-level re-programmable datapath or interprocessor communication structures are yet available. We are trying to find ways to realize high-level datapath and communication components using both laser and software-programmable on-chip interconnect and multiple buffering register files.

Several key issues have been identified for the datapath blocks: granularity of the processing elements (PE's), choice of the operator(s) for the PEs, interconnectivity between PEs,

communication of control flags between PEs and the controller, datapath widths, and I/O bandwidth. The prototype, dubbed PADDI for Programmable Arithmetic Devices for Digital Signal Processing, has been designed to cover a range (1-10) of sampling-interval-to-clock ratios to handle lowly multiplexed to highly multiplexed datapaths. PADDI contains 8-bit linkable execution units, a statically programmable hierarchical interconnect network, and a hierarchical control mechanism. The initial design envisions a central external controller chip that implements the control flow of the algorithms and broadcasts the state to different execution units, which may reside on different chips. Fast programmable state sequencers are commercially available for this purpose. A local nanostore determines the operations to be performed on a particular datapath. Both the nanostore and the configuration memory for the interconnection network can be loaded serially upon initialization. We anticipate that the nanostore will also be able to provide a next state address for branching. We are currently in the process of defining the micro-architecture and VLSI floor-planning for PADDI.

Because a hardware synthesis and verification environment will be crucial to the success of fast prototyping, this project also involves developing software tools to map a system definition into the hardware macrocomponent library and automatically generating functional and electrical simulation models.

2.19. HYPER -- An Interactive Synthesis Environment for High-Performance Real-Time Applications (J.M. Rabaey)

The goal of this project is to build an interactive environment for the design of high-performance real-time processors, consisting of the following major elements:

(1) The algorithm to be implemented must be specified either graphically or textually. We have extended Silage [1] to describe control constructs for register-level descriptions and developed a graphic front-end based on VEM/RPC.

(2) From the input specification, a mixed signal-flow/control-flow graph is derived. The undecorated flow graph, generated by the parser, is then passed to a scheduling and optimization pass. These techniques are described in [10].

(3) From the resulting scheduled graph, we can synthesize the datapaths, the controller, and the interface logic. All these steps require accurate information about the available cell library (speed, area, black-box view, and functionality); this information is provided by a rule-based library database. The final layouts of the processors are generated with the Lager IV system. A program that translates a decorated flow-graph description to the SDL (structure description language) has been developed.

The hardware mapping process consists of three routines that translate the datapaths, the controller, and the interface logic. The hardware mapping of datapaths requires a suite of transformation steps, including multiplexer reduction, hardware choices of assign operations, and data path partitioning. The details of these operations can be found in [3]. Several translation steps are also introduced to the system to handle signal broadcasting and arithmetic commutativity.

Datapath partitioning is based on three criteria. First, a depth first search is performed through the hardware graph. Each group is further divided if different word lengths are found within the group. If the number of blocks in a group is still too large, a simulated annealing-based algorithm is used for solving the min-cut problem.

The control path of a processor can also be derived from the decorated flow-graph. First, a state-transition diagram is generated from the scheduling information; then the transition diagram is optimized by removing the dummy states. The controller structure is generated in this step. Several optimizations are performed to reduce the size of the controller and to simplify the wiring between the control path and the datapath: these include recognizing the control signals that are independent of control states, merging equivalent or complementary signals, and allocating the minimum number of control registers by lifetime analysis.

Several extensions of the system have been made since last proposal. First, Control slices between datapaths and central control are partitioned according to the partitioning of the datapaths. This greatly improved the layout quality. Logic optimization is also performed on the control slices to reduce redundant logic. Another extension is that bus structure of control signals are allowed to handle cases such as log shifters and register files.

We have been able to generate the layouts of a 7th order IIR filter from Silage description and use THOR to simulate the functionality of the structure description generated by HYPER. Several examples, including the epsilon processor and the Viterbi processor of our Hidden Markov Model (HMM) speech projects, have also been generated from the flow-graph description. We can easily study the tradeoffs of different scheduling and hardware allocation schemes by using HYPER. Future research will involve the estimation and hardware allocation scheme of HYPER, introducing more translation and transformation routines, and rewriting the database system in OCT to improve performance.

2.20. High-Quality Speech Coding for Portable Communications (J.M. Rabaey)

This research focuses on the evaluation of high-quality speech coding algorithms and will result in an implementation for use in a portable communications system. The evaluation phase consists of analysis and simulation of several candidate algorithms. High-quality and relatively low bit-rate (~16 Kb/s) operation suggest a robust CELP (Code-Excited Linear Prediction) algorithm. In CELP coding, a search algorithm periodically selects an optimum excitation sequence from a vector quantized code-book. An index to this vector is then transmitted and used to excite a synthesis filter in the decoder. The decoder filter parameters are then updated in a backward adaptive fashion.

Since this coder must function in a portable environment, low power dissipation is an overriding concern. Suitability for a low-power, real-time implementation will undoubtedly influence algorithm selection and customization. Indeed, much of this study will focus on analyzing and developing VLSI architectures and design styles suitable for low-power operation. This analysis based on fundamentals of complexity theory, statistical analysis, and simulation should lead to design styles and methodologies suitable for low-power implementations. The research is relevant not only for this particular application, but also for a wide range of DSP algorithms

constrained to low-power operation.

2.21. Partitioning DSP Algorithms onto Multiprocessors with Configurable Interconnection (J.M. Rabaey)

Multiple programmable digital processors are used to handle the computationally intensive behavioral simulation of DSP algorithms. To minimize the communication overhead, the processors are connected by a bus that can be configured by software to match the communication pattern of each particular algorithm. To minimize the idling time, an efficient heuristic to balance processor loads is being investigated. This algorithm uses estimates of the computation time and communication time of the individual atomic tasks to optimally partition the program onto the processors. First, pipelining and retiming at the block level will be performed. Then, within the sub-block, parallelism or further pipelining can be used to exploit the finer grain concurrency.

The ultimate goal of this project is to provide an environment where users can express DSP algorithms textually or graphically, and have a compiler partition and translate the program into sections of assembly code to be executed on the multiprocessor system. The compiler will configure the bus to minimize the communication overhead.

A prototype of the multiprocessor system (SMART) has been built. It consists of 10 DSP32C Digital Signal Processors as well as custom VLSI chips to handle the communication and synchronization between processors.

The Silage To SMART (S2S) Compiler, which implements the partitioning algorithm, is under development. It is composed of 4 tasks:

1 Silage to Flowgraph (S2F) Translation

2 Flowgraph Partitioning

3 Flowgraph to C (FF2C) Translation

4 C to DSP32C Code Compilation

The Silage to flowgraph translation is complete. The flowgraph partitioning performs pipelining, retiming, and parallelism simultaneously under one global search strategy. The algorithm also automatically break the nodes of the graph to the appropriate level of granularity while partitioning. Initial results from the flowgraph partitioning are very good. Further research are needed in modeling communication costs between processors accurately, as well as in handling multirate DSP applications.

The first version of the flowgraph to C translation has been implemented, which can execute on Sun Workstations. The ultimate goal is to use AT&T's C compiler for the DSP32C to generate code for the processors. To yield efficient code, an in-depth study of the characteristics of the C compiler will be done.

2.22. SMART: Switchable Multiprocessor Arhitecture with Real-Time Support (J.M. Rabaey)

A major part of the design effort for DSP systems is devoted to algorithmic specification and verification. Executing these simulations on a general-purpose computer requires so much CPU-time that real-time simulations of certain algorithms are impossible. The main purpose of this project is to develop a dedicated simulation engine at least two orders of magnitude faster than a general-purpose computer architecture of the same technology level. To handle the number-crunching bottleneck, we are using a floating-point DSP processor (the DSP32C from AT&T Bell Laboratories) as the core processor. On this processor, simulations run at least an order of magnitude faster than on a general-purpose microprocessor.

An additional order-of-magnitude gain in simulation speed can be obtained by exploiting the high degree of parallelism and pipelining inherent in most signal-processing algorithms. We have proposed a multiprocessor architecture with a software-reconfigurable communication pattern that permits the processor architecture to be adjusted to match the concurrency and pipelining properties of the algorithm.

Two 208-pin VLSI chips have been designed and tested to handle communication and synchronization between the processors and to manage memory access. A prototype of the system is implemented and operational at peak 120 MFLOPS with 10 processing units. Application software programs such as diagnostic testing, image processing, and 1024-points FFT were developed to measure the performance of the architecture. Future efforts are directed to the developments of more application programs, the custom high-speed I/O interface, and the performance enhancement of the prototype system (200 MFLOPS) for same number of processors.

2.23. Extended THOR (J.M. Rabaey)

THOR is a functional simulator based on the CSIM simulator, a conventional event-driven functional/behavioral simulator. To describe a system in the THOR environment, the user has to provide the models of the system modules and their interconnections. The modules are written in a language called **CHDL** (C Hardware Description Language). **CHDL** is based on the C programming language with added features for hardware modeling. The interconnection network description is provided in a *component-oriented* net list language, called **CSL**.

THOR implements some of the features, which are essential in a heterogeneous simulation environment.

- Since the leaf modules in THOR are described in standard C, it is possible to link in a variety of "foreign" or external simulators, such as a SILAGE or a Motorola 68000 simulator.
- The THOR simulation engine handles the intermodule communications. Unfortunately, the present THOR is based solely on the event driven protocol, which is sufficient for structural simulation. As described above however, a system level simulation environment equally requires the data flow and communication processes based mechanisms.

The LAGER silicon assembly system makes extensive use of the THOR environment for modeling and simulation at the structural level. And the extension of the THOR environment, which will have most of the properties required for heterogeneous simulation, becomes essential.

- The CSL language will be replaced by the sdl/OCT-SIV descriptions of the LAGER environment. This will improve the parameterizability and the descriptive power.
- The introduction of foreign simulators will be simplified by providing a library of interface routines. This will make the interconnection mechanisms transparent to the user.
- The simulation engine will be adapted to incorporate a variety of communication mechanisms.
- The extended THOR will handle multi-processor simulation and *hard* simulation. This will once again be achieved by providing a set of library-routines, which hide the physical implementation of those interfaces to the user.

2.24. Behavioral Transformations for the Synthesis of High-Performance DSP Systems (J.M. Rabaey)

To solve a given computational problem, one can use a large number of algorithms. Often any one of these algorithms can lead to several implementations, each with vastly different time of execution, hardware requirements, power constraints, and testability. Since the flow graph specified by the designer often fails to meet the performance specifications or results in an inferior realization, optimizing transformations must be applied. Most of the behavioral transformations are well known from optimizing software compilers; they include constant arithmetic, common subexpression elimination, and dead-code elimination. More important are the loop transformations: loop retiming, loop pipelining, partial or complete loop unrolling, and loop jamming. These latter transformations are especially suitable real-time systems, in which each program contains an infinite loop of time and concurrency can be exploited more efficiently by controlling the hardware resources.

We are implementing a search-driven transformation environment where the order and type of the transformations are determined by the table of hardware-utilization ratios. The same environment can also support other high-level synthesis tasks such as module and clock selection, partitioning, pipelining, design-style selection, assignment, and scheduling.

We started implementation of transformations by implementation of basic block transformation. Those include commutativity, associativity, distributivity, retiming, constant evaluation, pipelining and software pipelining. We developed new algorithm based on Welsh randomized algorithm which apply just mention transformations in such way that we can schedule resulting signal flow graph in a minimum amount of time on a given hardware configuration. The first result (we completed implementation of commutativity) are very promising.

2.25. Scheduling and Resource Allocation in the Design of High-Performance Digital Signal Processor" (J.M. Rabaey)

The goal of this project is to minimize the total hardware cost of an implementation of a target program represented by a signal-flow graph (or data-dependency graph), given constraints on execution time, timing, and hardware. The hardware cost function is composed of the cost of the

functional units, memory, and connectivity.

An overview of the state of the art in this area is given in [1]. None of the available approaches, however, allows a consistent and unbiased treatment of the three contributions to the cost function: execution units, memory, and interconnect. Furthermore, most available techniques remove all hierarchies from the flow graph before scheduling: functions are expanded, and loops are unrolled, resulting in huge graphs for most problems. We have been developing a suite of algorithms that address these deficiencies.

The new technique has four parts: control strategy, estimation, assignment, and generation of the exact solution (scheduling).

We finished implementation of the new assignment and scheduling algorithm. From the algorithmic point of view there are three new features: (i) it is combination of probabilistic and classical constructive algorithms; (ii) the probabilistic algorithm is rejectionless; and (iii) the classical constructive algorithm is based on the idea of discrete relaxation. The simple extension is making incorporation of specific transformation (commutativity) simple and efficient.

We show excellent results on a number of examples. Running time for examples with up to a hundred nodes is less than 1 second. We introduce a broad class of test examples and the idea of estimation which makes assessment of scheduling and assignment algorithms easier and more realistic. Also, we demonstrated on those examples the simple and efficient approach of trading the speed of scheduling for the quality.

2.26. A Hardware Environment for Rapid Prototyping of DSP Systems (J.M. Rabaey)

The main goal of this project is to provide a rapid prototyping hardware environment for designers of DSP systems. The key part of the system that we envision is a dynamically configurable network with high communication bandwidth. In addition to data routing, the network also supports features such as broadcasting and merging data.

During the early phase of the design cycle, designers can simulate their algorithms by plugging processor boards into our system. The network integrates these processors together to form a multi-processor simulation engine which provides high computation throughput as required by many DSP algorithms. The network can be configured according to the communication patterns of the algorithms to reduce overheads due to inter-processor communication. Heterogeneous processors, e.g., DSP's and RISC's, can be used to suit the nature of the computation. As the system evolves, ASIC's replace certain sub-systems where high performance is desired while the rest of the system, such as some front-end processing, can still be emulated by the processors.

Our system shortens design time by providing high computation throughput and programmability throughout the development cycle so that designers can get fast feedback of performance and hardware problems. It also eases the integration of custom-designed hardware into the final system.

Problems that we have to address include system extensibility, I/O requirements of the system, standardization of the bus protocol and the design of the communication network.

2.27. Pulsar Signal Recovery (J.M. Rabaey)

Pulsars are rotating, highly magnetized neutron stars which emit sharp pulses with high stability of frequency. Pulsar timing has a number of applications. However, to achieve pulsar timing, the signals which have been dispersed by propogation through the interstellar medium, have to be passed through a de-dispersion filter on the receiver end. The coherent de-dispersion technique involves implementing the inverse interstellar medium transfer function in an FIR filter. However, to de-disperse the entire 100 Mhz bandwidth of the signal would require a FIR filter of an order of a million taps.

Since such a huge FIR filter is not feasable, the current research is in making an FIR filter which would implement a 1000 tap FIR filter which would de-disperse about 1 Mhz of the signal bandwidth. Currently, we are carrying simulations on word length effects on the signal recovery. Once the word lengths are decided upon, the de-dispersion filter would be converted to a VLSI chip.

2.28. Frigg: A Simulation Environment for Multiple-Processor DSP Hardware Development (E.A. Lee)

In the last six months we completed and terminated this project, the construction of design aids for systems using commodity programmable DSPs. We view this as a promising start on better integrated design environments that incorporate heterogeneous models of computation. In this case, we have interfaced the hardware design environment to Gabriel, which synthesizes assembly code for programmable DSPs. So the designer's view of the system simultaneously incorporates a dataflow representation of the signal processing application and a netlist representation of the hardware being designed to implement that application.

Many practical DSP problems require systems with multiple programmable DSPs. Developing multi-processor systems is complicated, and traditional development tools do not provide adequate support. While simulators exist for all programmable DSPs, there has been no ready way for developers to simulate the interaction between a programmable DSP and other digital hardware, or between multiple (possibly heterogeneous) DSPs. This lack of flexible simulation capabilities has often meant long delays in hardware development, and the postponement of software testing and debugging until after hardware prototypes are built.

We have designed and implemented a simulation environment that meets the needs of developer of multi-DSP systems. Our simulator, Frigg, uses the Thor simulator from Stanford University as the simulation substrate. Thor is used to simulate all hardware elements other than DSPs, and to simulate all interconnections between elements. Frigg integrates the capabilities of Thor with those of manufacturer-supplied DSP simulators. The X window system is used to provide a multi-window user interface which allows a user to easily select and interact with different processing elements. Frigg uses the interprocess communications (IPC) facilities of Unix to interface independent cooperating simulation processes, which can run on different hosts. The communication mechanism developed is general, and may be used in the future to interconnect a wide variety of simulators in a multi-mode simulation environment.

Frigg has been interfaced to Gabriel, a graphical data-flow programming system (with arbitrary granularity) for multiple programmable DSP systems. A designer can create a new architecture by specifying the Thor models and creating the netlist. Then those properties of the architecture that are relevant to code generation, such as the number of processors and the interprocessor communication mechanism, are summarized in a form readable to Gabriel. Gabriel can then take any application program and produce code for the new architecture. An attempt has been made to make the Gabriel core as independent as possible of details of the target architecture, but more work is warranted to get truly easy retargetability.

Finally, we have assembled a laboratory facility for experimentation with multiprocessor DSP hardware and software. Currently, a four-processor DSP56000-based system is running in the lab, and has been integrated with other lab resources. The prototype system was donated by Dolby Laboratories, a San Francisco company. The system has been interfaced to Gabriel, using the hardware description mechanism described above. We have demonstrated that Gabriel can automatically synthesize code for the system without requiring that the user have any architecture-specific knowledge.

2.29. Heterogeneous Hardware Targets (E.A. Lee)

The Gabriel software system is capable of running local simulations of a signal processing system on a workstation, or generating assembly code for real-time execution. Unfortunately, at this time, the designer must choose between these two modes of operation, rather than a more natural mode that would allow a mixed system that involves interaction between the workstation and the real-time DSP hardware. The next generation software environment (called Ptolemy, described separately) will support such mixed systems naturally by supporting cooperation between distinct scheduling models. The aim of this project is to determine how to use this new capability to automatically target heterogeneous hardware architectures. The first such target will be a workstation with a programmable DSP-based board on its bus. An application program will simultaneously specify the activity in the workstation and on the DSP board, and the interaction between the two, but to the system designer the interface will appear seamless. To begin this effort we have developed an interface between a VME-based development system for the Motorola DSP56001 and Sun 3 system. The interface has been developed using Gabriel pending sufficient capability in its replacement system. The limitations of the VME bus and Sun 3 workstation restrict the real-time transfer rate of our interface to somewhat less than 20,000 samples per second. To proceed to the next logical step, therefore, we have cooperated with a small local company that is producing a DSP56001 board that will reside on the S-bus of a Sun Sparc workstation. We are developing the appropriate device driver and will begin to design closely interacting applications that use both computation resources, the DSP and the workstation.

2.30. Ptolemy: A Non-Dogmatic Third Generation Simulation Environment (E.A. Lee and D.G. Messerschmitt)

We have implemented and had considerable user experience with two generations of DSP simulation environments, Blosim and Gabriel. Both environments used a block-diagram data-flow paradigm for the description of the algorithms. For the future we see the need for other computational models, such as discrete-event scheduling, mixed compile-time and run-time scheduling, or computational models based on shared-memory data structures. These are not supported very gracefully by Blosim or Gabriel. Most importantly, we see the need for a flexible simulation environment which is extensible to new computational models without re-implementation of the system.

This has led us to begin development of a new environment which uses object-oriented programming methodology. Our goal is to make it non-dogmatic, in the sense that the environment itself does not impose any particular computational model, and it is extensible to new models by simply adding to the system and not modifying what is already there. Further goals are to incorporate features that have been successful in Blosim or Gabriel, such as achieving modularity and reusability of user-programmed software modules, friendly graphical window interfaces, and code generation for target concurrent architectures rather than just simulation.

A first version of the system currently has a synchronous data-flow scheduler and an embryonic discrete-event scheduler. This system will thus immediately be capable of simulating combinations of signal processing and network simulations (such as in packet speech and packet video) and combinations of behavioral and hardware simulation. We are currently working on interfacing the system to the same graphical interface used by Gabriel, which uses Vem to edit Oct facets. Consequently, Oct serves as the design database for the system, just as with Gabriel.

REFERENCES

[1] P. A. Ruetz and R. W. Brodersen, "Architectures and Design Techniques for Real-Time Image-Processing IC's," *IEEE Journal Solid-State Circuits*, April 1987, p.233

[2] B. Richards, "Design of a Video Histogrammer Using Automated Layout Tools," UC Berkeley Electronics Research Laboratory, Memorandum No. UCB/ERL M86/38, 1986.

[3] See M. Srivastava, "Dedicated Real-Time Systems for Robotics," in this chapter of the *Research Summary*.

[4] T.A. Chu,"Synthesis of Self-Timed Control Circuits from Graph-Theoretic Specifications," *1987 IEEE International Conference on Computer Design*

[5] T. Amon, G. Borriello, C. Sequin, "Operation/Event Graphs and OESIM," *Technical Report #90-01-17*, Dept. of Computer Science and Engineering FR-35, Univ. of WA, Jan. 1990.

[6] T.H.Y. Meng, R.W. Brodersen, D.G. Messerschmitt, "Automatic Synthesis of Asynchronous Circuits from High-Level Specifications," *IEEE Transactions on Computer-Aided*

Design, Vol. 8, No. 11, Nov. 1989.

[7] P. Hilfinger, "QA High Level Language and Silicon Compiler for Digital Signal Processing", *Proc. Custom Integrataed Circuits*, Conf., Portland, OR, May 1985, pp. 213-216.

[8] M. Potkonjak and J. Rabaey, "Q:A Scheduling and Resource Allocation Algorithm for Hierarchical Signal Flow Graphs" *Proc. Design Automation Conference*, Las Vegas, NV. June 1989.

[9] C. Chu, et al., HYPER: An Interactive Synthesis Environment for "HYPER: An Interactive Synthesis Environment for High Performance Real Time Applications," *Proc. ICCD*, Boston, October, 19889.

[10 W. Koh, A. Yeungk, P. Hoang, and J. Rabaey, "A Configurable Multiprocessor System for DSP Behavioral Simulation," *ISCAS Symposium*, May, 1989.