MRASM: A Multi-Level Rule-Based Architectural Synthesis Methodology for Hierarchically Clustered Parallel Systems

Pao-Ann Hsiung
Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
E-mail: hpa@computer.org

Abstract

Although hierarchical *architectures are becoming more and more popular due to their high* scalability *and* performance, *yet there has been a lack of synthesis methodologies that are targeted at such hierarchically clustered systems. A novel methodology called* MRASM *is presented in this paper for the design of hierarchically clustered parallel systems. MRASM is a* multi-level rule-based *synthesis approach to parallel architecture design. The synthesized architectures at each level are simulated and checked for constraint satisfaction. An intelligent design space exploration method is incorporated into the simulation environment so that the target system is improved upon iteratively. One of the various alternative architectures is selected based on factors such as cost, throughput, utilization, bandwidth, scalability, reliability, and fault-tolerance. Experiments show that our approach is indeed feasible.*

Keywords: *hierarchically-clustered parallel systems, architectural synthesis, multi-level design, rule-based design, clustering technology*

1 Introduction

Scalability has been a major concern in the design of parallel computer systems. The design of hierarchical architectures with processors grouped into clusters and hierarchically interconnected with some interconnection network is one way to tackle the scalability problem. Parallel architectures with multiple processors have been the target of several recently published work on synthesis methodologies such as PSM [1], ICOS [2], POSE [3], and CMAPS [4], but they all do *not* consider hierarchical architectures which need specific design techniques such as partitioning the synthesis tasks into several levels of details. Here, a pioneer work on the design of a synthesis methodology targeted at hierarchical architectures, called *MRASM*, is presented.

Previous work on the synthesis of parallel systems do not explicitly consider hierarchical architectures. This leads to the possible waste of design time due to significant redundant efforts spent on synthesizing the same *cluster* design. Here, a hierarchical system consists of several clusters, where each cluster has a collection of interconnected processors, some memory, buffers, and control units. Usually clusters are homogenous such that only one cluster need be designed and the others are obtained by a simple reconfiguration of the first cluster. Existing synthesis methodologies will design each cluster as a separate new entity since they must consider the possibility of heterogeneity among the clusters. But, MRASM knows that hierarchical systems generally have homogenous clusters since heterogeneity often degrades overall system performance. Thus, MRASM is much more efficient at synthesizing hierarchical architectures than existing methodologies.

Section 2 describes the target Hierarchical Parallel System model. Section 3 gives an account of how MRASM synthesizes a system at different levels. An implementation example is given in Section 4. Section 5 concludes the paper.

2 Hierarchically Clustered Parallel System

The target system is a hierarchically clustered parallel architecture. *Hierarchically-Clustered Parallel System* (HCPS) has been gaining more and more importance [5, 6, 7, 8]. Some of the under-research and already-implemented systems are: the Cedar parallel processor at the University of Illinois at Urbana-Champaign [9], the Cm* at Carnegie-Mellon University [10], the Stanford University DASH architecture [11], the Syracuse University NETRA machine [12], and the MR-1 multiprocessor at the University of Melbourne, Australia [5].

The HCPS architecture consists of four parts: a *Global Control Unit* (GCU), a *Cluster Subsystem* (CS), an *Interconnection Network Subsystem* (IN), and a *Global Memory Subsystem* (GM). The GCU controls the data-loading of the system inputs into memory, and the partitioning and allocation of tasks to Clusters.

The control here can be one of two types: *Single Instruction Stream and Multiple Data Streams* (SIMD) or *Multiple Instruction Streams and Multiple Data Streams* (MIMD) [13]. The Cluster Subsystem consists of a set of totally homogenous clusters. Each cluster is composed of one or more *Processing Elements* (PE), one or more banks of *Local Memory* (LM), and the cluster interconnection network subsystem. Each PE is further composed of one *Central Processing Unit* (CPU) and a private memory. The Cluster Interconnection Network can be a shared bus, an indirect connection like multi-stage interconnection network, or a direct connection like hypercube or crossbar [14, 15]. Interconnection Network Subsystem is the main interconnection between GM and CS in the Shared Memory Architecture and an interconnection of clusters in the Message Passing Architecture. One or more banks of globally shared memory constitutes the Global Memory Subsystem in the Shared Memory Architecture. The type of memory access latency affects the global distribution of the memory banks.

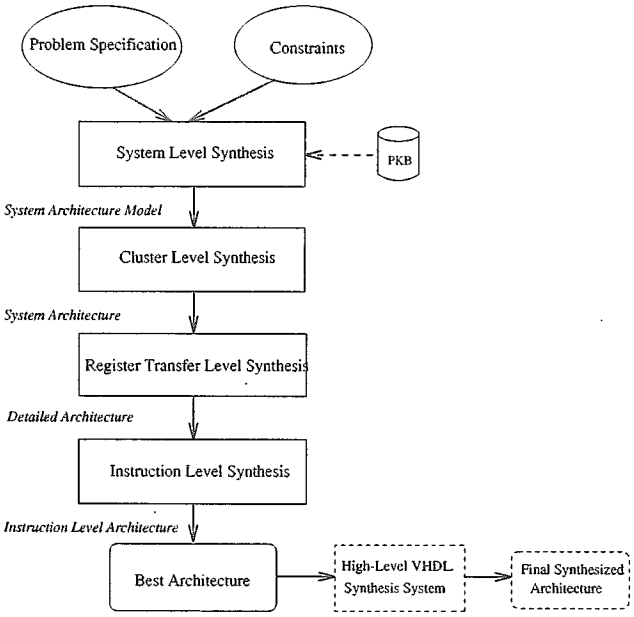

Figure 1: Multi-Level Synthesis

3 Multiple Levels of MRASM

MRASM is an architectural synthesis methodology for hierarchically-clustered parallel systems (HCPS). HCPS designers input their constraints and the problems to be solved. A Problem Knowledge Base (PKB) acts as a repository of known elementary problems such as sorting, searching, numerical computations, etc. Designers can choose problems from PKB for their system specification. The specification is input in the form of a directed graph with problems as vertices and data flow as edges. This graph form input has been used in several design methodologies [4]. Due to space-limitations, the reader is advised to refer to [4] for further details on the graph form input.

As shown in Figure 1, MRASM partitions system design into several levels: *System Level Synthesis* (SLS), *Cluster Level Synthesis* (CLS), and *Register-Transfer Level Synthesis* (RLS). Key design issues such as processor selection, memory selection, interconnection selection, and control selection are all considered in MRASM. The final VHDL [16] output can be used as input to a lower-level synthesis system, thus resulting in the parallel architecture being actually fabricated. Hence our approach is indeed a feasible one.

Rules are used to guide the synthesis process at each level of design. An intelligent *Design Space Exploration* (DSE) method is used to shorten the design-time and select the best architecture very efficiently. The target architecture is modeled and simulated using *SES/Workbench* [1] which is a simulation tool. Performance evaluation and analysis results obtained during simulation using this tool were input to the design space exploration process.

3.1 System-Level Synthesis

As shown in Figure 2, this first level of synthesis results in a system-level architecture model. The graph form input is mapped into a System Architecture Model. The mapping is performed in various stages: decision about the adoption of a Shared Memory or Message Passing Architecture, the selection of an appropriate System Memory Model, the selection of the type of System Interconnection Network, and the selection of the Control Method. This mapping is carried out by delving on the computation requirements, communication overhead, memory access patterns, and control dependencies as exhibited by the problem. All selections at the system level of synthesis are rule-based since there is not enough information for one to exactly analyse the effect of one's selections on the overall performance.

- *Shared Memory or Message Passing Architecture*: Heavy communication in a problem indicates the need for a Shared Memory type of Architecture as sharing of variables is more efficient than passing of messages. A problem with a small communication to computation ratio could be solved efficiently with a Message Passing loosely-coupled multicomputer system. It would seem that problems with

[1] SES/Workbench is a registered trademark of Scientific and Engineering Software, Inc.

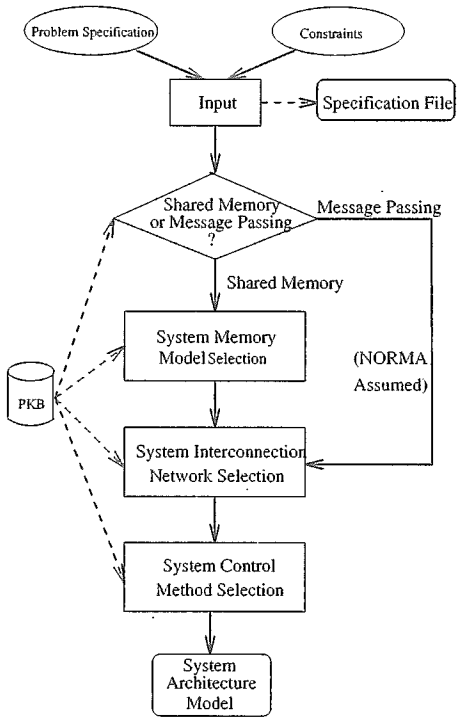

Figure 2: System Level Synthesis

roughly equal fraction of communication and computation could be mapped onto either type of architecture, it is here that user-given constraints come into play, for example, a high architecture scalability requirement on the part of the user would definitely make us opt for the Message Passing Architecture since loosely-coupled systems are more easily scalable than tightly-coupled ones.

- *System Memory Model Selection*: Different types of memory models are taken into consideration: *Uniform Memory Access* model (UMA), *Non-Uniform Memory Access* model (NUMA), *Cache-Only Memory Access* model (COMA), and *NO-Remote-Memory Access* model (NORMA) [17]. A problem showing uniform memory access pattern would indicate the need for an UMA Model, whereas one exhibiting non-uniformness in memory accesses would certainly indicate a NUMA Model. A problem that shows very little but does need memory sharing would prefer a COMA Model. The NORMA Model is reserved for Message Passing Architecture.

- *System Interconnection Network Selection*: As mentioned in Section 2, System Interconnection Network can one of *Shared Bus* (SB), indirect connection like *Multi-stage Interconnection Network* (MIN), or direct connections like HyperCube or Crossbar. Low scalability requirements, few processing nodes (Clusters), and low cost would indicate the use of SB as interconnection network, whereas high cost and high performance would require something like crossbar as interconnection network. Moderate requirements would indicate something intermediate to SB and Crossbar in cost and performance, which is in fact the MIN.

- *System Control Method Selection*: Certain problems like matrix-computations are more suited for computation using an SIMD method of control whereas there are problems which require an MIMD control method. Hence, the method of control can be chosen at this level of synthesis. If a problem can be decomposed into similar tasks (performing the same operations) then an SIMD method of control is more suitable as only data varies among the tasks. If a problem when decomposed, consists of dissimilar tasks then an MIMD method of control would be more suitable.

3.2 Cluster-Level Synthesis

Figure 3 shows the different stages at this level of synthesis. Here we input the System Architecture Model obtained from the previous level through four stages of Cluster Level Synthesis: *Processor Selection*, *Cluster Memory Model Selection*, *Cluster Interconnection Network Selection*, and *Cluster Control Selection*.

The last three stages of this level of synthesis are almost the same as those of the last level, differing only in that they are considered in a more local sense, i.e., within a Cluster.

- *Processor Selection*: Processor Selection depends on the computation requirements and the degree of parallelism of the given problem as well as the cost constraints imposed by the designer since processors are the most expensive component of the system. High performance processors are usually more expensive than low performance ones; given an upper limit on the cost expenditure that could be incurred on the whole system, one must not only consider what type of processors to choose (high or low performance, expensive or cheap ones), but also the number of processors to use per Cluster and the number of Clusters. Heavy computations call for the need of high performance expensive processors, and a given cost constraint would permit us to use only a small number of them, but if that same problem also shows a high degree of parallelism in computation, one may opt

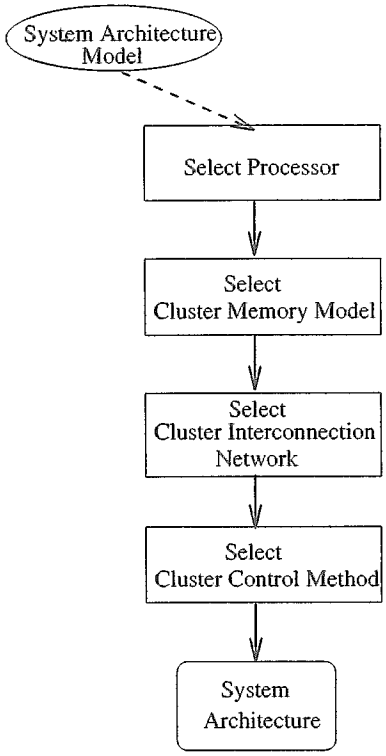

Figure 3: Cluster Level Synthesis

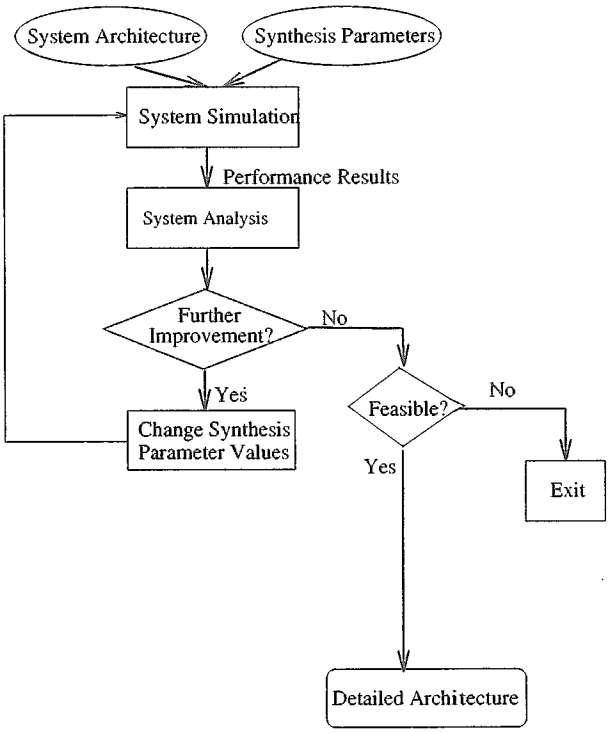

Figure 4: Register-Transfer Level Synthesis

for some not-so-high performance processors so that we may have a greater number of processors at our disposal to solve the problem more efficiently and with a better throughput. The result of this level of synthesis is a System Architecture which is in fact a simple virtual parallel machine.

3.3 Register-Transfer Level Synthesis

See Figure 4 for an overview of Register-Transfer Level Synthesis. As shown in Figure 5, the DSE used in this approach is a multi-level one. The main iterative part of *Design Space Exploration* (DSE) is contained in this level of synthesis. The System Architecture obtained from the Cluster Level Synthesis is one of the inputs at this level, the other inputs are the *Synthesis Parameters* and their values. The synthesis parameters and the performance factors are as given in Table 1 and Table 2, respectively. System simulation is required to explore the design space at this level. System analysis is the process of selecting the synthesis parameter to change in the next DSE iteration.

3.3.1 System Simulation

The system architecture as obtained from the previous level is verified and refined through simulation and DSE, respectively. Workloads pertaining to the user-given problem are input to the system simulator and performance results are recorded.

Initially, the Synthesis Parameters are given values corresponding to the resulting System Architecture from the System Level and Cluster Level Synthesis. These values in fact form the configuration of the system at iteration level 0. System is then simulated with workloads representing the user-given problem. Performance results, e.g., throughput, utilization are recorded. System Analysis determines the synthesis parameter to change in the next iteration based on these results, and accordingly the system is re-simulated to re-record the performance results of the improved system. Thus the system is simulated once each iteration.

3.3.2 System Analysis

As shown in Figure 6, the primary objective of System Analysis is to determine which synthesis parameter to change so that the overall system performance is improved. The effect of changing each synthesis parameter on each performance factor is approximated using first-order B-spline piece-wise linear segments [18].

Performance Space (PS) It is defined as an n-dimensional real space with dimension i representing P_i, $i = 1, 2, \ldots, n$

Consider the iterative process of DSE. Let r be the current iteration number, $r = 0, 1, 2, \ldots$; S_j be the j^{th}

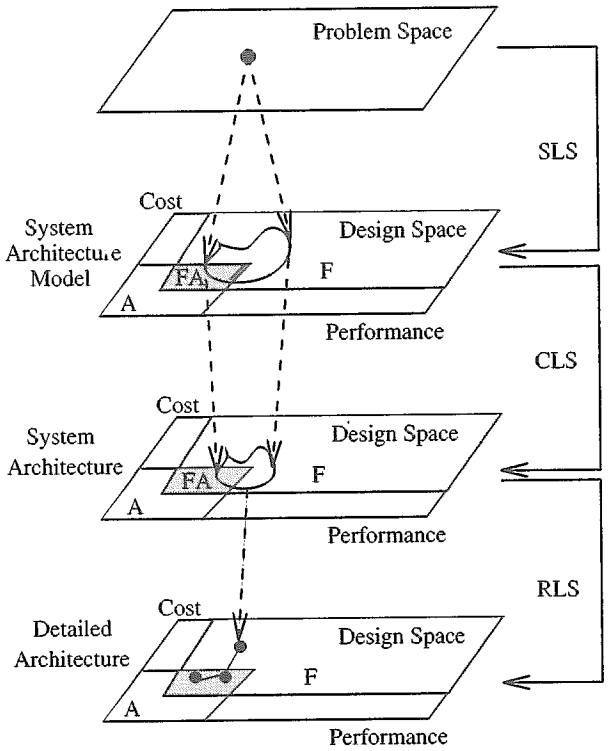

Figure 5: Multi-Level Design Space Exploration

Table 1: Synthesis Parameters

	Synthesis Parameters
1	System Memory Model (*UMA, NUMA, COMA, NORMA*)
2	Type of System Interconnection (*SB, MIN, Hypercube, Crossbar*)
3	Size of system interconnection (*no. of bus, size of MIN, dimension of cube*)
4	Number of Clusters
5	Number of PE per cluster
6	Method of Global control (*SIMD, MIMD*)
7	Kind of processors (*fine/coarse grained, RISC/CISC*)
8	Type of cluster interconnection (*SB, MIN, Hypercube, Crossbar*)
9	Size of cluster interconnection (*No. of bus, size of MIN, dimension*)
10	Method of Cluster control (*SIMD, MIMD*)

Table 2: Synthesis Performance Factors

	Perf Factors	Definitions
1	Cost	Total system component costs.
2	Power	Ratio of Throughput to Utilization
3	Reliability	$\Pi(1-(1-R_a(t))^{x_a})$
4	Fault-Tolerance	$\sum_{a=1}^{n} f_a \times \frac{d_a}{x_a}$
5	Scalability	$\sum_{a=1}^{n} f_a s_a$

R_a = the reliability of component a. d_a = maximum number of allowable faulty component a, f_a = importance factor for component a, and s_a = scalability of component a.

Synthesis Parameter, $j = 1, 2, 3, \ldots, m$; and P_k be the k^{th} Performance Factor, $k = 1, 2, 3, \ldots, n$.

DSE Algorithm: *Iterate*: $(r = 0.1, 2, \ldots)$

1. Approximate
   the effect of each S_j on each P_k, k=1,2,...,n, j=1,2,...,m. Let $f_k[Synthesis(S_j)]$ be a function that returns the value of P_k on changing S_j by one unit. Hence, we need to approximate

$$E_{j,k} = \left(\frac{\delta f_k[Synthesis(S_j)]}{\delta S_j} \right), \quad (1)$$

   where $k = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$. Let $X_{j,k} = \frac{\Delta P_k}{\Delta S_j}$ be a random variable. Stochastically, obtain the mean and variance of $X_{j,k}$. Use $(\mu_{j,k}(x_{j,k}), \sigma^2_{j,k}(x_{j,k}))$ as an approximation of $E_{j,k}$. This in fact is a B-Spline piece-wise linear approximation.

2. Calculate expected design point on changing S_j, $j = 1, 2, \ldots, m$. Suppose this is the $(r+1)$th iteration of DSE. Let I_r be the design point obtained in the r^{th} iteration. Note that I_0 is the design point obtained as a result of first two levels of synthesis. Let $E_j(I_{r+1})$ be the expected location of design point in the next iteration due to a change in S_j. That is, $\forall j, \forall r$, $E_j(I_{r+1}) = I_r + \sum_{k=1}^{n} \vec{\mu}_{j,k}$, where $r = 0, 1, \ldots$, $j = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n$. Here, $\vec{\mu}_{j,k}(x_{j,k})$ is considered to be a vector parallel to the k-axis in PS.

3. Select Synthesis Parameter to change:

   Case 1 : $\forall j, \exists k$ such that $[E_j(I_{r+1})]_k > P_{k_{max}}$, where $[E_j(I_{r+1})]_k$ is the kth co-ordinate and $P_{k_{max}}$ is the upper bound on P_k.

   $Norm_{j,k} =$

$$\begin{cases} \left(\frac{[E_j(I_{r+1})]_k}{P_{k_{max}}} - 1 \right)^2 & \text{if } [E_j(I_{r+1})]_k > P_{k_{max}} \\ 0 & otherwise. \end{cases} \quad (2)$$

   Case 2 : $\forall j, \forall k, [E_j(I_{r+1})]_k \leq P_{k_{max}}$, let $Norm_{j,k} = \left(\frac{[E_j(I_{r+1})]_k}{P_{k_{max}}} \right)^2$ and $\forall j, Dist_j = \sqrt{\sum_{k=1}^{n} \alpha_k Norm_{j,k}}$, where α_k is the weight

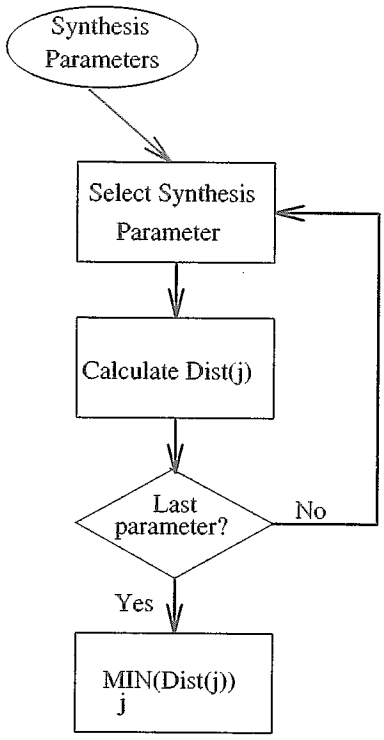

Figure 6: System Analysis

of P_k. If $Dist_p = Min_j(Dist_j)$, let $Dist(I_{r+1}) = Dist_p$, **return**(p).

4. Change S_p by one unit, keeping the others constant.

5. **If** (# of iteration = limit) or $(Dist(I_{r+1}) - Dist(I_r)) \leq \varepsilon$, **then** exit **else** goto step (1).

Hence the design is iteratively improved by moving the design point at closer and closer to the best architecture. If the final design point lies within the Feasible and Acceptable Region of Performance Space then the final detailed design architecture is output, otherwise the synthesis system exits with error messages.

4 Implementation

SES/Workbench, a modeling/simulation tool has been used throughout the synthesis process to model and simulate the whole system. This tool has also been used to translate the SES models into VHDL models. Figure 7 shows the main Multiprocessor System Model with workloads in the form of control jobs, computation jobs and data-loads. GCU and Cluster models are omitted here due to page-limit. The behavior of a PE is illustrated in Figure 8.

Example: An example multiprocessor system has been synthesized using the proposed synthesis approach

Table 3: Various CPU and their features

Company CPU	Intel 80486	Intel Pentium	HP PA[3]	DEC Alpha	AMI[2] PPC
SPECint92	32	64.5	80	130	117
SPECfp92	16	56.9	150	193.6	242.4
MHz	33	66	99	200	66.5
Cost ($)	317	900	N/A	$\approx$ 1000	$\approx$ 2500
Type	CISC	CISC	RISC	RISC	RISC
Model	—	—	735	21064	*
[2]AMI=Apple-Motorola-IBM, [3]PA=Precision Architecture, PPC=PowerPC, *=MPC601					

Problem : sorting, searching, and matrix multiplication.
Constraints: $c(n) = O(n \log n)$, $n \approx 10^8$
CPU benchmark : $SPECint92 \geq 100$
$SPECfp92 \geq 100$
Total cost $\leq \$110\,000$

1. **SLS**:
   (a) Shared Memory or Message Passing?
   • High Communication to Computation Ratio. $\Rightarrow$ Use Shared Memory.
   (b) UMA, NUMA, OR COMA?
   • No non-uniformity in memory access $\Rightarrow$ UMA.
   (c) System Interconnection?
   • Shared Bus or MIN.
   (d) System Control Method?
   • SIMD (since sorting, searching, and matrix multiplication can be partitioned into similar tasks; e.g. subsequence sorting)

2. **CLS**:
   (a) Processor Selection: Referring to Table 3, only DEC's Alpha chip and Apple-Motorola-IBM's PowerPC meet the SPECint92 and SPECfp92 constraints.
   (b) Cluster Memory Model: UMA.
   (c) Cluster Interconnection: Shared Bus or MIN.
   (d) Cluster Control: SIMD

3. **RLS**: Some assumptions ...
   • 1 Shared Bus cost = $100 (capacity:10)
   • 8x8 MIN cost = $100
   • 1 Bank of 4 MB Memory Cost = $150
   • No. of Mem. Banks in GMS = No. of Clusters
   • No. of Mem. Banks in CMS = No. of PE/Cluster

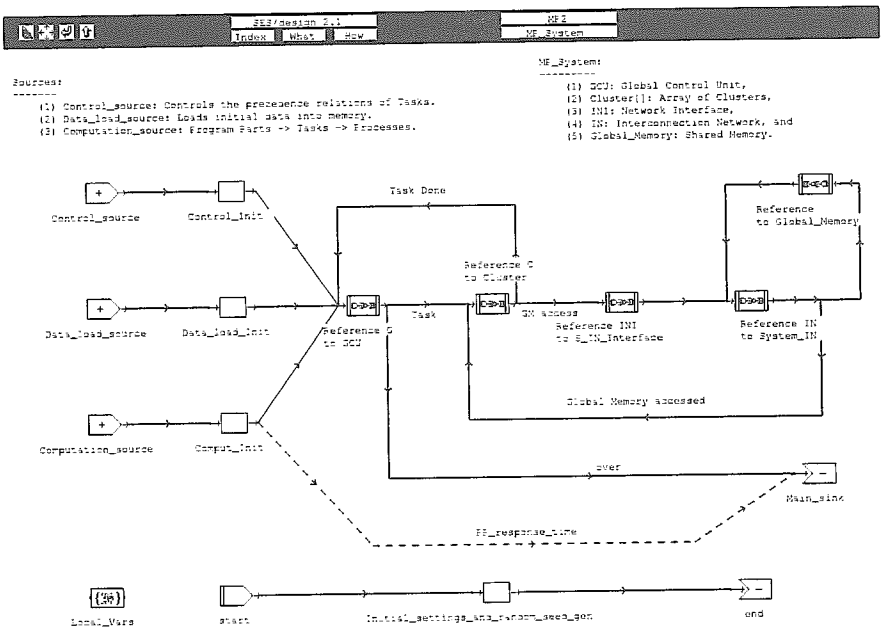

Figure 7: SES Model: Overall System Architecture Model

- Cluster Memory Cost = $ 500
- Control Circuitry + Others = $ 500

On Simulation and DSE, we deduct the following results:

- 5 Clusters gives good performance.
- 1st alternative: 100 Alpha chips with one 8x8 MIN costs $103,350.
- 2nd alternative: 40 PowerPC with one Shared Bus costs $103 350.
- Further analysis and simulation shows that the 1st alternative is a better one due to its higher fault-tolerance and scalability.

5 Conclusion and Future Work

We have presented a novel methodology which is a pioneer work on synthesizing hierarchically-clustered parallel systems. MRASM is a multi-level rule-based approach to synthesizing such systems. Experiments conducted using MRASM have shown significant increase in synthesis efficiency as compared to existing systems that do not take advantage of the hierarchy in such systems. An intelligent DSE algorithm integrated into the simulation/refinement/validation process of sythesis is also an important feature of MRASM. Future directions include more system design experiences and a formal verification model [19] for such system designs.

References

[1] P.-A. Hsiung, S.-J. Chen, T.-C. Hu, and S.-C. Wang, "PSM: An object-oriented synthesis approach to multiprocessor system design," *IEEE Trans. on VLSI Systems*, vol. 4, pp. 83–97, March 1996.

[2] P.-A. Hsiung, C.-H. Chen, T.-Y. Lee, and S.-J. Chen, "ICOS: An intelligent concurrent object-oriented synthesis methodology for multiprocessor systems," *ACM Trans. on Design Automation of Electronic Systems*, vol. 3, pp. 109–135, April 1998.

[3] P.-A. Hsiung, "POSE: A parallel object-oriented synthesis environment," *ACM Trans. on Design Automation of Electronic Systems*, vol. 7, April 2002. (to appear).

[4] P.-A. Hsiung, "CMAPS: A cosynthesis methodology for application-oriented parllel systems," *ACM Trans. on Design Automation of Electronic Systems*, vol. 5, January 2000. (to appear).

[5] S. A. Mabbs and K. E. Forward, "Performance analysis of MR-1, a clustered shared-memory multiprocessor," *Journal of Parallel and Distributed Computing*, vol. 20, pp. 158–175, 1994.

[6] F. D. Kubler and F. Lucking, "A cluster-oriented architecture for the mapping of parallel processor

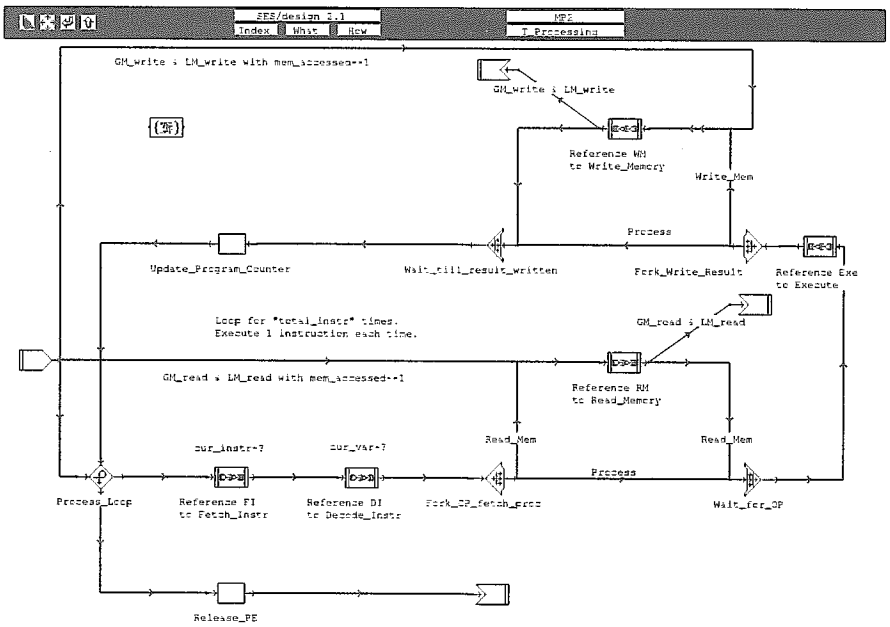

Figure 8: SES Model: Processing Element

networks to high performance applications," in *In Proc. of the 1988 Int. Conf. on Supercomputing*, July 1988.

[7] A. J. Wilson, "Hierarchical cache/bus architecture for shared-memory multiprocessors," in *In Procs. of the 14th Int. Symp. Computer Architecture*, pp. 422–433, 1987.

[8] K. B. Irani and A. R. Naji, "Performance analysis of a clustered shared-memory multiprocessor," in *In Proc. of the 1988 Int. Conf. on Supercomputing*, July 1988.

[9] D. Gajski, D. Kuck, D. Lawrie, and A. Sameh, "Cedar: A large scale multiprocessor," *ACM Comput. Arch. News*, vol. 11, pp. 7–11, March 1983.

[10] R. J. Swan, S. H. Fuller, and D. P. Siewiorek, "Cm*: A modular multimicroprocessor," in *In Proc. of the AFIPS National Computer Conference*, pp. 637–644, 1977.

[11] D. Lenoski, J. Laudon, K. Gharacholoo, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, "Design of the stanford dash multiprocessor," in *Technical Report CSL-TR-89-403, Computer Systems Lab, Stanford Univ.*, December 1989.

[12] A. Choudhary, "Performance of vision algorithms on multiple clusters in NETRA," in *In Procs. of the 1990 Int. Conf. on Parallel Processing*, 1990.

[13] M. Flynn, "Some computer organization and their effectiveness," *IEEE Trans. Computers*, vol. C-21, pp. 948–960, September 1972.

[14] C. Seitz, "The cosmic cube," *Communication of the ACM*, vol. 28, pp. 22–33, January 1985.

[15] H. Sullivan and T. Bashkow, "A large scale homogeneous, fully distributed parallel machine, i," in *In Proc. of the 4th Symp. on Computer Architecture*, pp. 105–117, March 1977.

[16] *IEEE standard VHDL Language Reference Manual*, February 1992.

[17] K. Hwang, *Advanced Computer Architecture*. New York: McGraw-Hill, Inc., 1993.

[18] A. J. Gadient and D. E. Thomas, "A dynamic approach to controlling high-level synthesis CAD tools," *IEEE Trans. on VLSI Systems*, vol. 1, pp. 328–341, September 1993.

[19] P.-A. Hsiung, T.-Y. Lee, and S.-J. Chen, "MOB-net: An extended Petri net model for the concurrent object-oriented system-level synthesis of multiprocessor systems," *IEICE Trans. on Information and Systems*, vol. E80-D, pp. 232–242, February 1997.