Contents

Parallelization of Iterative Heuristics for Performance-Driven Low-Power VLSI Standard Cell Placement

Sadiq M. Sait, Abdul Waheed, and Mahmood R. Minhas
College of Computer Sciences & Engineering
King Fahd University of Petroleum & Minerals
KFUPM # 673, Dhahran-31261, Saudi Arabia
e-mail: {sadiq,awaheed,minhas}@ccse.kfupm.edu.sa

Abstract

The inherent complexity involved in VLSI design and its subproblems has always made them ideal application areas for iterative heuristics. However the major drawback has been the large runtime involved in reaching acceptable solutions especially in the case of multi objective optimization problems. Among the acceleration techniques proposed, parallelization of these heuristics is the most promising. In this project we develop parallel algorithms, techniques and tools for these heuristics, namely Simulated Annealing, Tabu Search, Genetic Algorithms, Stochastic Evolution, and Simulated Evolution applied on multi objective VLSI cell-placement problem. These placement objectives are to reduce power dissipation and wirelength while improving performance (delay). This parallelization is to be achieved on a cluster of PCs interconnected by a low-latency network.

Keywords: VLSI physical design, Iterative Algorithms, Optimization, Low-Power, Performance, Placement, Parallel Heuristics, Parallelization, Multiobjective, Metaheuristics, Cluster Computing.

1 Introduction and Motivation

With the advances in VLSI technology, it is possible to achieve fabrication on the nano-technology scale with component densities of millions of transistors per circuit. Due to its complexity the VLSI design process is divided into several intermediate levels of abstraction. More details about the new design are introduced as the design progresses from higher to lower levels. Typical levels of abstraction, together with their corresponding design steps, are illustrated in Figure 1.

Until recently, physical design optimizing the wirelength (area) and performance (timing) was of prime importance. Results of several research efforts to optimize area and performance are available in literature and summarized in several papers and books [1, 2, 3, 4, 5, 6]. Nowadays, with several new power-constrained applications ranging from mobile phones to laptop computers, power dissipation has emerged as another important objective of VLSI circuit design [7]. The problem of optimizing the three above mentioned objectives in VLSI circuit design can be addressed at any of mentioned design steps. In this work, we are concerned with optimization at the physical level, in particular the cell placement phase.

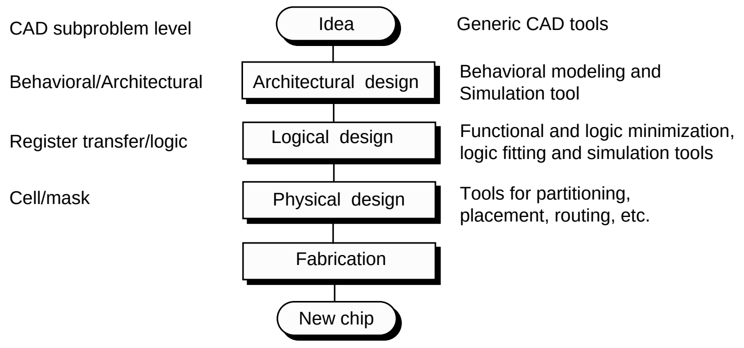

Figure 1: Levels of abstraction and corresponding design steps.

Achieving such multi objective optimization with the sheer complexity of modern circuit density is a NP-hard problem for which conventional constructive techniques have often proved inadequate. Iterative heuristics on the other hand have been a tremendous success in reaching acceptable solutions for such problems. A brief introduction of the more popular and accepted iterative heuristics employed in VLSI design is presented in the following section. For detailed studies, the reader is referred to [2].

1.1 Iterative Heuristics

The primary advantage of iterative heuristics over conventional constructive algorithms is their probabilistic ability to escape from local optima. Among the earliest of these heuristics, Simulated Annealing (SA) was proposed in the early eighties [8] and enjoyed phenomenal success when applied to combinatorial optimization problems from a variety of disciplines. SA is often credited for sparking the research interest in iterative algorithms and is considered a comparison benchmark for new heuristics in its class. It has become customary that every newly reported randomized search heuristic has to prove itself by performing better than SA on a number of (benchmark) test cases. The term *annealing* refers to heating a solid to a very high temperature and then slowly cooling the molten material thereby allowing the atoms to regain proper crystal structure. Simulated Annealing emulates this cooling behavior by allowing *temperature* based stochastic acceptance of bad solutions thereby escaping a probable local optima. At high temperatures the acceptance probability is almost unity and hence SA behaves as a randomized search algorithm. But as temperature decreases the acceptance probability for poor solutions is also reduced with SA behaving as a greedy algorithm at very low temperatures. SA is a very robust heuristic and given sufficient run-time can guarantee reaching an acceptable solution.

Genetic algorithms were invented in the early 1970s [9] and emulate the natural process of evolution to perform an efficient and systematic search of the solution space to progress towards the optimum. It is based on the theory of *natural selection* that assumes that individuals with certain strong characteristics are more able to survive, and hence pass these characteristics to their offsprings. GAs combine information exchange along with *survival of the fittest* among individuals to conduct the search. These algorithms operate on a *population* (or set) of *individuals* (or solutions) encoded as strings. These strings represent points in the search space. In each iteration, referred to as a generation, a new set of strings that represent solutions (called offsprings) is created by crossing some of the strings of the current generation [10]. Occasionally new characteristics are injected to add diversity.

Another modern iterative heuristic, called Tabu Search is based on the systematic exploration of memory functions. It is an aggressive search technique where for a given solution, a large number of neighbors are generated, from which the best is chosen. To determine which of the generated solutions is the best, an *evaluator* that is based on the objectives being optimized and the historical information that has been accumulated, is used [11, 12, 13]. The trace of the current solution is controlled by a recent move history to avoid cycling in the solution space. Use of intensification/diversification in Tabu Search considerably helps in obtaining superior quality solutions. This is accomplished with the help of additional memory structures that keep record of information such as frequencies of moves, elite solutions, etc. [2].

Simulated Evolution (SimE) is an elegant and general randomized iterative search heuristic. It adopts the generic state model where a solution is seen as a population of movable elements. Each element i is characterized by a **goodness** measure $g_i \in (0, 1)$. A goodness near 1 indicates a highly fit individual. Starting from a given initial solution, SimE repetitively executes the following steps until stopping conditions are met: Evaluation where the *goodness* is determined for each individual i of the population P. The goodness measure must be a single number expressible in the range $[0, 1]$; Selection which takes the population P together with the estimated *goodness* of each individual i as input and partitions P into two disjoint sets, a selection set P_s and a set P_r of the remaining members of the population; and finally the Allocation step where the two sets P_s and P_r are taken as input and a new population P' which contains all the members of the previous population P is generated. The elements of P_s are mutated according to an allocation function *Allocation*. From the results reported in the literature, it is observed that the SimE algorithm is a sound and robust randomized search heuristic. It is guaranteed to converge to a global optimum if given enough time and has modest runtime and space requirements.

Stochastic Evolution (StocE) is another powerful general and randomized iterative heuristic for solving combinatorial optimization problems [14]. StocE improves on SA by avoiding the initial random search. Good moves, i.e., moves which improve the solution are always accepted while bad moves may also get accepted with a non-zero probability. If there is no change in the solution state i.e., the search is trapped in local optimum, the acceptance probability for bad moves is increased. If there is an improvement in the best solution reached, StocE rewards itself with more iterations. The word *evolution* is used in reference to the alleged evolution processes of biological species.

1.2 Motivation for Parallel CAD

Despite the advances in VLSI technology, there are still a few challenges that pose an obstacle in its rapid development. One of them is the large run-time required for iterative heuristics which play a crucial role in VLSI design. Of the various acceleration strategies attempted, parallel computing has always exhibited the most potential. Not only is it possible to achieve shorter runtimes with parallel processing but also handle larger problem sizes, obtain better quality results, etc. These potential advantages are enumerated and detailed below [15]:

1. **Faster Runtimes:** Most of the VLSI design problems are computationally intensive and take a large amount of time ranging from several hours to days. Moreover, future design tools will require even more computational capabilities. Given such increased requirements for speed and accuracy, parallel processing is the only way to accelerate the design tasks.
2. **Larger Problem Sizes:** Sometimes, due to time or memory limitations, these design tools cannot handle larger problem sizes. This can be overcome by using parallel processing, as both computational speed and memory size are enhanced by using parallel architectures.
3. **Better Quality:** As most of the VLSI design problems are NP complete [16], heuristics used to solve them may give non-optimal solutions. The solutions obtained are function of the fraction of the search space traversed. With the use of parallel search techniques, better quality results can be obtained. This is possible as a larger search space can be traversed in the same time constraint.
4. **Cost-effective Technology:** With the use of low-cost microprocessors connected by some interconnection network, it is possible to harness computational power comparable to supercomputers but at the fraction of the price.

1.3 Parallel Processing Models

In this section some of the available parallel processing models and architectures are discussed. There exist two well-known parallel architectures for implementing parallel processing models. General purpose multiprocessors in which each processor can access a common shared address space are classified as *shared memory multiprocessors.* As compared to this model, in a *distributed memory multiprocessor* each processor has its own local memory.

The shared memory model allows all processors to access data from a shared memory without any explicit declarations. A high-speed bus serves as the interconnection medium between these processors. In distributed memory model, data is shared between the memories of individual processors through explicit message passing. Here we discuss the latter since it is most suitable to our present work.

The distributed memory models have evolved into cluster computing systems, Networks Of Workstations (NOWs) and signify co-operative computing using complete stand-alone workstations connected through dedicated low-latency networks. Although such clusters comprising general-purpose workstations provide an affordable and cost-effective means of generating enormous computing power, the challenge lies in achieving their effective utilization. In these parallel computing systems, communication between nodes (workstations) places a significant overhead on actual runtime due to the network latency. Hence inter-node communication has to be minimized. Effective partitioning of the computational load among the workstations is another important aspect, to minimize overlap between individual work-domains. Hence harnessing the computational power of cluster systems through efficient parallel programming is inarguably a significant challenge.

In this proposal, we discuss some generic and specific parallel algorithms for iterative heuristics applied to multi objective VLSI cell placement. Parallel implementations are achieved over a dedicated computing cluster with individual nodes connected via a low-latency network. The performance of these parallel algorithms in terms of run-time and search efficiency are made with their respective sequential counterparts running on a single machine.

This proposal is organized as follows: In Section 2, some theoretical aspects of parallelizing iterative heuristics are discussed along with a review of related literature. Section 3 deals with problem formulation and the proposed approach. Objectives, project design, schedule, management, utility value of the project, and detailed budget are given in the subsequent sections.

2 Review of Parallel Iterative Heuristics

In this section, a detailed survey of parallel iterative heuristics for VLSI standard cell placement is reported. A generic intuitive strategy for achieving parallelization is to partition the data into small subsets distributed among the processors [17, 18]. Each processor is responsible for a data subset and implements a sequential version of the concerned heuristic over this data subset.

However, for combinatorial optimizations, three types of parallelization strategies seem to be appropriate [19]. They are

1. The operations within an iteration of the solution method are parallelized.
2. The search space (problem domain) is decomposed.

3. Multi-search threads with various degrees of synchronizations are used.

More details on the choice and application of these parallel strategies to the iterative heuristics are discussed in the subsequent paragraphs. The iterative heuristics are:

- Simulated Annealing (SA)
- Genetic Algorithms (GA)
- Tabu Search (TS)
- Simulated Evolution (SimE)
- Stochastic Evolution (StocE)

2.1 Parallelization of Simulated Annealing (SA)

Several attempts on parallelizing SA can be found in the literature for small global memory multiprocessor systems [17, 20, 21] as well as for small distributed memory multiprocessors systems [18].

In most cases, parallelization is done at data level. Data, describing the problem, is split into small subsets distributed among the processors [17, 18]. Each processor is responsible for a data subset and performs sequential simulated annealing on it.

There are several problems involved with this kind of parallelization; the efficiency depends directly on the degree of dependence between different data subsets. High dependencies result in intensive communication and low efficiency. The structure of the subsets is closely related to the given optimization problem. Furthermore, the number of efficiently usable processors is strongly affected by the problem size and by the nature of the subsets.

In order to parallelize SA, the following general approaches have been applied [2].

1. Move acceleration also known as single trial parallelism: In this approach, each individual move is evaluated faster by breaking up the task of evaluating a move into subtasks and allocating various subtasks to different processors. This approach is also called move decomposition.
2. Parallel moves also known as multiple trial parallelism.

In move acceleration, a trial is generated and evaluated faster by distributing the various tasks on several processors working in parallel. For parallel moves, several trials are generated and evaluated in parallel, where a single processor executes each trial. The speed-up that can be achieved by single-trial parallelism approach largely depends on the particular problem and hence has to be tailored accordingly [2]. Most of the work on parallelization of simulated annealing has targeted the multiple-trial parallelism approach or a combination of the single and multiple trials approach.

Parallel evaluation within moves (Single trial parallelism)

In this approach, each individual move is evaluated faster by breaking up the task of evaluating a move into subtasks and allocating various subtasks to different processors. This approach is also called move decomposition.

Kravitz and Rutenbar [22] have proposed two different approaches to move decomposition. In object decomposition, the responsibility of moving a set of objects is delegated to a particular processor. While in functional decomposition, the individual portions of work such as wire-length evaluation and updating are delegated to different processors. The object and functional decomposition can be further divided into static and dynamic cases.

It should be noted that this approach to parallelization is limited to shared memory algorithms only and have limited scope for parallelism [15].

Non-Interacting Parallel Moves (Multiple trial parallelism)

In this approach, each move is applied on one processor, so that moves can be generated and evaluated simultaneously. The use of non-interacting parallel moves was proposed and implemented by Kravitz and Rutenbar [22], but accepting only a serializable set of moves. Suppose a set S of moves is evaluated in parallel. Let S_n be a subset of non-interacting moves in S. By concurrently applying all the moves to the same initial configuration, the same final configuration is obtained by applying moves in any order. Such a set of moves is said to be serializable. In order to gain maximum parallelism, a maximum set of serializable moves has to be determined, which itself is a hard problem.

There are several alternatives to choose from. One such heuristic is that only one of the acceptable moves is really accepted. Some potential parallelism is sacrificed using this approach. Another strategy would be that each processor chooses moves, independently from the entire set of available moves. A third strategy is to bias each processor's move choices using a spatial decomposition where each processor is assigned to perform moves within a region [22].

Figure 2 is a possible parallel simulated annealing algorithm following this multiple-trials parallelism approach. In Figure 2, it is assumed that one master processor is ordering the concurrent execution of p trials, where p is the number of processors. The master evaluates the outcome from all trials. In case of no success, the master then orders the parallel evaluation of p new trials; oth-

```
Algorithm Parallel_SA;
        (*$S_0$ is the initial solution *)
Begin
        Initialize parameters;
        $BestS = S_0$;
        $CurS = S_0$;
            Repeat
                Repeat
                Communicate $CurS$ to all processors;
                ParFor each processor $i$
                    Perturb($CurS, NewS_i$);
                    $A_i$ = Accept($CurS, NewS_i$) (* $A_i$ is true if $NewS_i$ is accepted *)
                EndParFor
                If Success Then
                    (* Success = ($\bigvee_{i=1}^{p} A_i = True$) *)
                    Select($NewS$);
                    If $Cost(NewS) < Cost(BestS)$ Then $BestS = NewS$;
                EndIf
                Until Time to update parameters;
            Until Time to stop;
            Output Best solution found
End. (*Parallel_SA*)
```

Figure 2: General parallel simulated annealing algorithm where synchronization is forced after each trial.

erwise, it selects the best new current solution among the accepted solutions, and updates the state of all processors. This process repeats until it is time to stop. Also at the end of each p new trials, the master processor checks to see whether equilibrium has been reached at the current temperature. If yes, then the algorithm parameters are updated.

2.2 Parallelization of Genetic Algorithms (GA)

The Genetic Algorithm is highly amenable to parallelization. This is a direct consequence of the parallel nature of the genetic reproductive processes. Most GA parallelization techniques exploit the fact that GA works with a population of chromosomes (solutions) and proceed as follows. The population is partitioned into subpopulations which evolve independently using sequential GA. Interaction among the smaller communities is occasionally allowed. This strategy exposes the explicit parallelism of GA and makes it a very suitable approach for running on a multi-processor machine. This parallelization approach has been shown to converge faster to desirable solutions. This parallel execution model is a more realistic simulation of natural evolution in which communities are isolated but, occasionally interact through migration or cross communities matings.

The reported parallelization strategies fall into three general categories: (1) the *island model*, (2) the *stepping stone model*, and (3) the *neighborhood model*, also called the *cellular model*. [2]

Island model

In this variant, the population is divided among the available processors. Each processor runs sequential GA on its local subpopulation. Periodically, subsets of elements are migrated among subpopulations. Migration is allowed between all subpopulations. Examples of implementations of this model are reported in [23, 24]. The island parallelization model is suitable for multiprocessor machines with a relatively small number of processors.

Stepping stone model

This strategy is similar to the *island model* except that communication is restricted to neighboring populations only [25]. This model is inspired by the theory of punctuated equilibria which has been suggested to solve certain paleontological dilemmas in the geological record [26, 27]. The theory says that species tend to stay stable as long as the environment is steady but, when there

```
Algorithm ($Parallel\_GA$)
    Every processor will have one subpopulation of size $p$.
    Initialize.
    For $j = 1$ to E Do
        ParFor
            Run GA for a fixed number of iterations;
            Send a subset of local subpopulation to neighbors;
            Add the received subsets to local subpopulation;
            Select $p$ elements from local subpopulation
        EndParFor
    EndFor;
    Return highest scoring element
End
```

Figure 3: Parallel GA using the stepping stone model.

is an environmental change, species undergo a rapid evolution to adapt to the new condition. In GA terminology, the model defines fitness of individuals relative to other individuals in the local subpopulation. This corresponds to having a steady environment. Each local community will stabilize after a number of generations. Then, by introducing new elements (new competitors) from neighboring communities, a change in the environment is simulated. The subpopulation elements will rapidly evolve to adapt to this new change. This parallelization approach is summarized in the algorithm of Figure 3. Examples of parallel implementations of this variant are reported in [25, 28, 29].

Neighborhood or cellular model

Although the above two models are reasonable parallel implementations, they are not suitable enough for massively parallel systems. Cellular genetic algorithms are designed to take advantage of massively parallel machines [30]. In this model, we throw away the local subpopulations boundaries and assume that communities are continuous. That is, every individual has its own neighborhood defined by some diameter. This model is more suitable for massively parallel systems, where each processor is assigned one element. The basic operations performed by each processor are summarized in the algorithm of Figure 4.

To reduce communication overhead, cellular GAs usually restrict mating based on distance. This form of restriction leads to what is called *isolation-*

```
Algorithm (Parallel_GA)
    Initialize.
    For $j = 1$ to E Do
        ParForeach processor
            Send cost of element to all neighbors;
            Select the neighbor with lowest cost;
            Perform crossover between local element and selected element;
            Replace local element with offspring according to some strategy
        EndParFor
    EndFor;
    Return highest scoring element
End
```

Figure 4: Parallel GA using the neighborhood model.

by-distance [31].

2.3 Parallelization of Tabu Search (TS)

The first reported studies on the parallelization of Tabu Search were published in the early 1990s [32, 33, 34].

In [35], the authors classify parallel tabu search heuristics based on a taxonomy along three dimensions.

- The first dimension is **Control cardinality**, where the algorithm is either *1-control* (One processor executes the search and distributes tasks to other processors.) or *p-control* (Each processor is responsible for its search and communication with other processors).
- The second dimension is **Control and communication type**, where the algorithm can follow a *rigid synchronization (RS)*, and *knowledge synchronization (KS)* (Synchronous operation mode where the process is forced to establish communication and exchange information at specific points explicitly defined), a *Collegial (C)*, and a *Knowledge Collegial (KC)* strategy (Asynchronous operation modes where contents of communication are analyzed, concerning the global characteristics of good solutions and search strategy).
- The third dimension is **Search differentiation** where the algorithm can be *SPSS (single point single strategy), SPDS (single point different strategies), MPSS (multiple point single strategy), or MPDS (multiple point*

different strategies).

Now based on this taxonomy, the previous tabu search algorithms have been categorized by [35]. In order to solve the flow shop sequencing problem by [33], a mechanism for parallel implementation of tabu search algorithm used search space decomposition strategy. It is a *1-control, RS, SPSS algorithm.* Another parallelization of tabu search for vehicle routing problem by [34] also uses search space decomposition strategy. It is also *1-control, RS, SPSS* algorithm.

In order to improve parallel tabu search using evolutionary principles, the algorithm presented in [32] used *multi-search thread* strategy. It is a *p-control, C, MPSS* algorithm. Another parallel tabu search algorithm for the 0-1 multi-dimensional knapsack problem was put forth by [36], which used *multi-search threads* strategy. It is a *p-control, RS, MPDS* algorithm.

In [37], a parallel tabu search algorithm for voltage and reactive power control in power systems was proposed. Of the two schemes, one of them used the *domain decomposition strategy*, while the other scheme followed a *multi-search threads* strategy. The first one is *1-control, RS, SPSS* and the second one is *p-control, RS, MPDS* algorithm.

Synchronous Parallel Implementation:

In this implementation a master process and one of the slave processes are running on one machine. The remaining slave processes are running on distinct machines. All slave processes are started with the same initial solution. After searching its part of the current neighborhood, each slave process reports its best move back to the master. The master process selects the best among the received best moves (subject to tabu conditions). If the stopping criteria are met then the search stops; otherwise the master broadcasts the selected move back to the slaves and the search continues.

The main steps of the reported parallel algorithm are outlined in Figure 5. For a more detailed description of this parallel implementation the reader is referred to [34] where a noticeable improvement in solution quality over one of the best constructive algorithms for vehicle routing problem, with substantial reduction in runtime, is reported.

Asynchronous Parallel Implementation:

In this approach, each processor explores a subset of the neighborhood of its current solution. Each processor is competing with its neighbors (its adjacent

```
Algorithm ParallelTabuSearch;
Begin
1.    Construct initial solution and broadcast it to all slaves;
2.    Each slave process explores its own neighborhood and
      sends its best move to the master;
3.    The master identifies a subset of moves that improve the
      objective function (subject to tabu restrictions);
4.    If there is no such move Then the master randomly selects
      one uphill move;
5.    The master broadcasts to all the slaves the selected move(s);
6.    Each slave performs the received move(s) and makes necessary updates;
7.    If time-to-stop Then Return BestS;
8.    Goto Step 2
End.
```

Figure 5: Synchronous parallel implementation of tabu search.

processors) in finding a superior solution. When the stopping criteria are met, every processor reports its best solution. The general outline of this parallelization approach is given in Figure 6.

Similar asynchronous parallel tabu search implementations for the traveling salesman and quadratic assignment problems have been reported in [32]. The results reported indicate a marked improvement in solution quality as well as convergence speedup. In [32], the time between successive solution exchanges is called *diffusion interval*. The authors stated that the best results were obtained by performing this exchange and updating current solution at each iteration as indicated in steps 5 and 6 of Figure 6.

2.4 Parallelization of Simulated Evolution (SimE)

Unlike other iterative heuristics, parallelization of Simulated Evolution (SimE) has not been the subject of much research. The only effort at parallelizing Simulated Evolution was by the inventors of the technique [38, 39].

The basic approach to parallelizing the algorithm involved (for the VLSI cell placement problem) assigning a subset of the physical placement grid to each processor. The partitioning was done in a row-wise because this greatly simplifies the adaptation of the algorithm to the parallel environment and avoids adverse effects created by unequal cell lengths. A set of rows is mapped

```
Algorithm PeerProcess;
Begin
1.    Construct initial solution and initialize parameters;
2.    Explores own neighborhood;
3.    Select best move subject to tabu restrictions;
4.    Update tabu list;
5.    Exchange current best solution with neighbors;
6.    Update current solution based on received neighbor solutions;
7.    If time-to-stop Then Return best solution;
8.    Gotostep 2
End.
```

Figure 6: Asynchronous parallel implementation of tabu search.

to each processor; it is assumed that the number of rows is much greater than the number of processors, at least three to four times greater. Two different partitioning patterns are sufficient to ensure correct operation of the algorithm on any number of processors. The pattern for each processor alternates every iteration to ensure that each cell can move to any position on the grid in atmost two steps, as long as the number of rows per processor is sufficiently large. The left pattern shows the distribution for odd-numbered iterations and the right one, the partitioning for the even-numbered cycles.

Each processor performs a complete iteration of the evolution-based algorithm on the subset of cells assigned to it and on its assigned regions on the chip. During a subsequent communication phase, the processors update their placement information by combining all individual results to the new common placement. Then, the whole process repeats itself, starting with the assignment of new partitions. Figure 7 shows the outline of the algorithm. [15]

It should be noted that the preceding parallel algorithm (for VLSI cell placement problem) is effective for cases where the number of processors is much less than the number of rows. When the number of processors is equal or greater than the number of rows, a different parallel algorithm needs to be designed. In such a case, the basic strategy is to partition the cells and associated net among the processors. A copy of the entire chip layout is replicated on each processor. A processor is responsible for evaluation, selection and allocation of the cells that are owned by it. After a single iteration of evaluation, selection and allocation of cells, the processors update the global state information of the layout and resolve any inconsistencies through realignment of

```
Algorithm Parallel_SimE;
    Start with an initial placement of cells;
    Partition the chip area among processors by rows
    Ownership of cells by processors owning chip subregion;

    REPEAT
        FOR ALL processors in PARALLEL DO
            Broadcast current cell positions to all processors;

            Start partitioning pattern 1
            FOR each cell in the chip DO
                Evaluate goodness of cell using cost function;
            FOR each cell in the chip DO
                IF random(0,1) > goodness of cell;
                    THEN select cell, place in queue for new allocation;
            FOR each cell in allocation queue DO
                Allocate cell in empty locations among own region;
            Broadcast current cell position to all processors;

            Start partitioning pattern 2
            FOR each cell in the chip DO
                Evaluate goodness of cell using cost function
            FOR each cell in the chip DO
                IF random(0,1) > goodness of cell
                    THEN select cell, place in queue for new allocation;
            FOR each cell in allocation queue DO
                Allocate cell in empty locations among own region;
        END FORALL
    UNTIL (termination condition);
```

Figure 7: Parallel Algorithm using Simulated Evolution with Domain Partitioning. (For VLSI cell placement problem)

the cells in various rows. The global synchronization is performed during the realignment phase by as many processors as there are in the rows, with each processor responsible for each row. After that, each process will broadcast the latest correct state of its rows to all processors. The cycle of evaluation, selection, allocation and update will be iterated. [15]

2.5 Parallelization of Stochastic Evolution (StocE)

Unlike other heuristics, like GA and SA, Stochastic Evolution is highly sequential. Because of this, the easiest parallelization approach would be to proceed as follows. Each processor is assigned a particular initial solution. Then each of the processors would be running sequential StocE starting from its assigned initial solution. This simple approach would be very good if the search subspaces of the various processors do not overlap (or have minimal overlap). In this case all processors would be concurrently searching distinct parts of the solution space. However, this would require that one has enough knowledge about the search space in order to partition it among the individual processors. In some instances, this can be a very unrealistic assumption, because very little will be known about the search space. Diversification and intensification ideas from Tabu Search may be used to partition search space.

For many problems, the subspace corresponding to the neighborhood of a particular solution is controlled by the algorithm designer (the state model, the parameter p, and the move operation). In many cases, it may be possible to tune the algorithm for a particular problem instance so that the state space is searched in parallel (with minimal overlap) by several processors.

To parallelize stochastic evolution, we can follow similar approaches to those adopted for the parallelization of simulated annealing, namely: (1) *move acceleration*, and (2) *parallel moves*. In move acceleration, a move is performed faster by distributing the various trial relocations on several processors working in parallel. The speed-up that can be achieved by *move acceleration* approach depends to a large extent on the problem instance. Each simple move usually consists of several trial relocations of a particular movable element. The trial relocations can be performed in parallel without affecting the correctness of the algorithm. For problem instances where the window of trial relocations is large, sizable speed-up can be achieved.

For *parallel moves*, several moves are performed in parallel, where each move is executed on a single processor[1]. Figure 8 is a general description of

[1]This approach assumes that the moves do not have to be performed in any specific order. This parallelization strategy will not work if the moves must be attempted in a fixed

a possible parallel stochastic evolution algorithm following this parallelization strategy.

In the description of Figure 8, it is assumed that one master processor is ordering the concurrent execution of p simple moves, where p is the number of processors. The master evaluates the outcome from all trials. In case of no success, the master then orders the parallel evaluation of p new trials; otherwise, it selects the best new current solution among the accepted solutions, and updates the state of all processors. This process repeats until it is time to stop. At the end of the parallel execution of *PERTURB*, the master processor may be required to run the *MAKE_STATE* procedure to ensure a valid new state. The master processor is also in charge of updating the parameter p and the counter ρ.

The parallel algorithm given in Figure 8 is a synchronous parallelization where the processors are forced to communicate and synchronize after each trial. However, since the current solution will get updated only when a processor accepts a move and changes the state, the various processors should be allowed to proceed asynchronously with their trials until at least one of them accepts a simple move. Therefore, one can improve the parallel algorithm of Figure 8 by making the following change. The movable elements are distributed equally among the available processors. Each processor will be in charge of the trial relocations of its associated movable elements. Synchronization is forced only when one of the processors performs a successful trial. In this new variation communication will be less. Furthermore, it is a more efficient parallelization since no processor is forced to remain idle waiting for other processors with more elaborate trials to finish. [2]

Both variations of this parallel algorithm can be implemented to run on a multicomputer or a multiprocessor machine. The parallel machine model assumed is an MIMD machine. For both algorithms, it is assumed that each processor must be able to set a common variable to *True* whenever it accepts a simple move; then the solution accepted by the processor is communicated to a master processor which will force all other processors to halt and to properly update the current solution. The current solution together with the remaining unprocessed elements are again distributed among the available processors. Unfinished moves, i.e., moves that were in progress when their processors were forced to halt will be tried again in the next iteration. Another possibility is to allow the processors to complete the trials that are in progress when the request to stop was received. Then there could be more than one solution

predetermined order.

```
**Algorithm**Parallel_StocE;
        /* $S_0$ is the initial solution */
**Begin**
        Initialize parameters;
        $BestS$=$S_0$; $CurS$=$S_0$; $p = p_0$;
              **Repeat**
                    Communicate $CurS$ and a movable element $m_i$ to each processor $i$;
                    **ParFor** each processor $i$
                          $NewS^i$=MOVE($CurS$, $m_i$);
                          **If** Gain($CurS$, $NewS^i$) $> RANDOM(-p, 0)$
                               THEN $A_i$ = TRUE;
                    **EndParFor**
                    **If** Success **Then**
                          /* Success = ($\bigvee_{i=1}^{p} A_i = True$) */
                          Select($NewS$); /* $NewS$ is best solution among all $NewS^i$'s */
                          **If** Cost($NewS$) = Cost($CurS$) **Then** $p = p - 1$;
                          **Else** $p = p_0$;
                          **EndIf**
                          **If** Cost($NewS$) < Cost($BestS$) **Then**
                               $BestS$= $NewS$;
                               $\rho = \rho - R$
                               **Else** $\rho = \rho + 1$;
                          **EndIf**
                    **EndIf**
              **Until** $\rho > R$;
              **Return** ($BestS$)
**End**. /*Parallel_StocE*/
```

Figure 8: General parallel stochastic evolution algorithm where synchronization is forced after each trial.

accepted, and therefore the master processor has to arbitrate between them, select the best, and pass a copy to each processor. All moves tried during this iteration, whether accepted or rejected are considered complete. During the following iteration only the remaining unprocessed movable elements will be tried.

Another possible approach would be to allow the processors to proceed concurrently with their search and to concurrently accept moves, with no interaction whatsoever. Algorithms following this strategy are known as *error algorithms*. The word error is used to highlight the fact that the processors have incorrect knowledge about the state of the parallel search. Few studies on such parallelization approach for the case of simulated annealing have indicated, that by limiting the number of concurrent moves or by ensuring that the moves are always *non-interacting*, error is minimized and convergence is maintained [40, 41, 22]. However, it is not always clear how one can go about restricting the moves to be of a particular type.

3 Problem and Cost Functions Formulation

In this project, parallel iterative heuristics for VLSI standard cell placement will be designed and implemented with the objective of optimizing power dissipation, delay and wire length.

3.1 Problem Statement

The cell placement problem can be stated as follows: Given a collection of cells or modules with ports (inputs, outputs, power and ground pins) on the boundaries, the dimensions of these cells (height, width, etc), and a collection of nets (which are sets of ports that are to be wired together), the process of *placement* consists of finding suitable physical locations for each cell on the entire layout. By suitable we mean those locations that minimize given objective functions, subject to certain constraints imposed by the designer, the implementation process, or layout strategy and the design style. The cells may be standard-cells, macro blocks, etc. In this work, we deal with standard cell design, where all the circuit cells are constrained to have the same height, while the width of the cell is variable and depends upon its complexity [2].

3.2 Cost Function for Power

In standard CMOS technology, power dissipation is a function of the clocking frequency, supply voltages and the capacitances in the circuit.

$$p_{total} = \sum_{i \in V} p_i(C_i \times V_{dd}^2 \times f_{clk}) \times \beta \tag{1}$$

where p_i is the switching probability of gate i over a clock cycle, C_i represents the capacitive load of gate i, f_{clk} is the clock frequency, V_{dd} is the supply voltage, and β is a technology dependent constant. Assuming that the clocking frequency and power voltages are fixed, the total power dissipation of the circuit is a function of the total capacitance and the switching probabilities of the various gates in the logic circuit. The capacitive load of a gate comprises the input gates capacitances of cells and those of interconnects.

$$C_i = \sum_{j \in F_i} C_j^g + C_{ij}^r \tag{2}$$

where C_j^g is the capacitance for gate j, C_{ij}^r represents the interconnect capacitance between gates i and j, and $F_i = \{j|(i,j) \in E\}$. Two other terms contribute to power dissipation, the short-circuit current and the leakage current. These are not considered at this level of design.

3.3 Cost Function for Delay

A digital circuit comprises a collection of paths. A path is a sequence of nets and blocks from a source to a sink. A source can be an input pad or a memory cell output, and a sink can be an output pad or a memory cell input. The longest path (*critical path*) is the dominant factor in deciding the clock frequency of the circuit. A critical path makes a problem in the design if it has a delay that is larger than the largest allowed delay (period) according to the clock frequency.

The delay of any given path is computed as the summation of the delays of the nets $v_1, v_2, ..., v_k$ belonging to that path and the switching delay of the cells driving these nets. The delay of a given path π is given by

$$T_\pi = \sum_{i=1}^{k-1} (CD_{vi} + ID_{vi}) \tag{3}$$

where CD_{vi} is the switching delay of the driving cell and ID_{vi} is the interconnection delay that is given by the product of the load factor of the driving cell and the capacitance of the interconnection net, i.e.,

$$ID_{vi} = LF_{vi} \times C_{vi} \tag{4}$$

$SLACK_\pi$ of path π is given by

$$SLACK_\pi = LRAT_\pi - T_\pi \tag{5}$$

where $LRAT_\pi$ is the latest required arrival time and T_π is the path delay [42, 43]. If T_π is greater than $LRAT_\pi$, then the path π will have a negative $SLACK$ which is an indicator of a **long path** problem. Upper bounds can be applied to nets belonging to the critical path as constraints not to allow them to exceed a certain limit beyond which the $SLACK$ will be negative.

In this work, we shall use the approach reported in [42] to predict the K-most critical paths. The placement program will seek to satisfy the delay constraints imposed by these paths.

3.4 Cost Function for Wirelength

Different models have been proposed for the estimation of length of a given *net*. Semi-perimeter of bounding box, minimum Steiner tree, minimum spanning tree, etc., are among those models [2, 44]. A Steiner tree approximation described below, which is fast and fairly accurate in estimating the wire length will be adopted in this work [45]. To estimate the length of net using this method, a bounding box, which is the smallest rectangle bounding the net, is found for each net. The average vertical distance Y and horizontal distance X of all cells in the net are computed from the origin which is the lower left corner of the bounding box of the net. A central point (X, Y) is determined at the computed average distances. If X is greater than Y then the vertical line crossing the central point is considered as the bisecting line. Otherwise, the horizontal line is considered as the bisecting line. Steiner tree approximation of a net is the length of the bisecting line added to the summation of perpendicular distances to it from all cells belonging to the net. Steiner tree approximation is computed for each net and the summation of all Steiner trees is considered as the interconnection length of the proposed solution.

$$X = \frac{\sum_{i=1}^{n} x_i}{n} \qquad Y = \frac{\sum_{i=1}^{n} y_i}{n} \tag{6}$$

where n is the number of cells contributing to the current net.

$$Steiner\ Tree = B + \sum_{j=1}^{k} P_j \tag{7}$$

where B is the length of the bisecting line, k is the number of cells contributing to the net and P_j is the perpendicular distance from cell j to the bisecting line.

$$Interconnection\ Length = \sum_{l=1}^{m} Steiner\ Tree_l \tag{8}$$

where m is the number of nets.

In standard cell placement, cells (or blocks) of fixed heights are placed in rows. It is the width of these rows that varies with the proposed solution according to the type and number of cells placed in the row. An approximation would be to treat cells as points, but in order to estimate lengths of interconnects more accurately, widths of cells are taken into account. Heights of routing channels are estimated using the vertical constraint graphs constructed during the channel routing phase. With this information, a fairly accurate estimate of the overall area, delay, and power dissipation can be obtained.

4 Project Objectives

The proposed research work will build upon our previous research that focused on design and engineering of sequential iterative algorithms for VLSI standard cell placement. The iterative algorithms implemented include SA, GA, TS, SimE, and StocE. The parameters optimized include power dissipation, performance (delay) and wire length while layout width was taken as a constraint.
The primary objective of the present work is to design and implement parallel iterative heuristics for multi objective VLSI standard cell placement. Some other objectives of the work are enumerated below:

1. The current project is expected to advance research into parallelization of iterative heuristics for multiobjective optimization problems.
2. The present work will also provide an insight into the various design phases where multi objective optimization can be performed. It is expected that optimization at other levels of abstraction will also become apparent, and the techniques designed in this work can be extended to those levels.
3. The overall outcome of this work will be the development of efficient cell placement tools for high-performance and low-power VLSI physical design, which can be integrated into existing design automation systems.

5 Tasks Outline

The preliminary work currently being carried out includes setting up experimental environment, configuration of cluster of workstations, and porting of the previously implemented sequential code from Windows platform to Unix/Linux. Following the preliminary work, the high-level tasks comprise analysis of the sequential implementations to identify bottlenecks and study various strategies and parallelization models that would best overcome these bottlenecks. This analysis will be followed by implementation and performance evaluation of the parallel heuristics.

The details of the major tasks to be carried out during the project work can be enumerated as follows:

Task 1: Setting up a Network Of Workstations (NOWs) as a cluster environment for study and implementation.

Task 2: Review and analysis of the sequential implementations of iterative heuristics to identify the bottlenecks in terms of CPU time, memory requirements and I/O requirements. Hence these bottlenecks are amenable to parallelization.

Task 3: Literature review of recent work related to parallelization of iterative heuristics for the VLSI cell placement problem as well as for other optimization problems.

Task 4: Investigation of different parallelization strategies for iterative heuristics in relevance to the VLSI cell placement problem.

Task 5: Collection of data and tools for implementation, analysis and performance evaluation of the proposed parallel model.

Task 6: Design and implementation of parallel iterative heuristics viz. GA & TS.

Task 7: Design and implementation of parallel iterative heuristics viz. SA, SimE & StocE.

Task 8: Compilation of the results and comparison of parallel and sequential implementations in terms of their convergence rates, quality of solution as well as run-times and memory requirements.

Task 9: Documentation of the developed software.

Task 10: Generation of periodical project reports and authoring of publications for conferences/journals.

6 Schedule

The work schedule is shown in the table below. All investigators and graduate students will be involved in all the tasks of the project for the entire duration.

	(**Months**)							
	01-03	04-06	07-09	10-12	13-15	16-18	19-21	22-24
Task 01	—							
Task 02	—	—						
Task 03	—	—	—	—				
Task 04	—	—	—					
Task 05		—	—	—				
Task 06		—	—	—				
Task 07		—	—	—	—	—	—	
Task 08				—	—	—	—	
Task 09					—	—	—	
Task 10		—		—		—	—	—

Management Plan

The project team will consist of a principal investigator, two co-investigators, and four part-time graduate students. Responsibilities will be divided among the three senior investigators on the basis of their previous experience and background.

Dr. Sadiq M. Sait has major interests in VLSI Design automation, and, in engineering and applications of computers. He has published several papers in the area of VLSI physical design automation. He has co-authored two books, which are directly related to the project: (a) *VLSI Physical Design Automation: Theory and Practice*, McGraw-Hill Book Co., Europe, December 1994. Also Co-published by **IEEE Press**, USA, January 1995 (Hard bound edition), and, (b) *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems.* December 1999, IEEE Computer Society Press, California.

Dr. Abdul Waheed is an assistant professor in Computer Engineering Department at KFUPM. Before joining COE, he was working at Inktomi Corporation in Foster City, California, USA as a performance engineer in network products division. He was a research staff member at NASA Ames Research Center, Moffett Field, California, USA from May 1997 until July 2000. He held a summer position in Concurrent Computing Division at Hewlett-Packard Research

Laboratories in Palo Alto, California, USA in 1994. He received the BSc degree with honors in Electrical Engineering from University of Engineering and Technology, Lahore, Pakistan in 1991. He received the MS degree in 1993 and the PhD degree in 1997, both in Electrical Engineering from Michigan State University, East Lansing, Michigan, USA. His current research interests include performance evaluation, high-performance parallel computing & networking systems, and multimedia systems. He has written over twenty refereed conference and journal papers on related topics. Dr. Waheed is a member of the IEEE Computer Society.

Mahmood R. Minhas is currently working as a lecturer in Information & Computer Science (ICS) department at KFUPM. Before joining ICS, he worked as a Research Assistant in COE department at KFUPM. He received the MSc degree in Computer Science from Int'l Islamic University, Islamabad, Pakistan and MS degree in Computer Engineering from KFUPM in 2001. His current research interests include VLSI design automation, iterative algorithms and multi objective optimization. He has recently published three papers in the area of VLSI physical design automation.

All investigators will take the responsibility of the overall management of the project. Students will be computer engineering/science graduates with good programming background and will work under the guidance of investigators. The investigators will hold meetings as often as necessary to coordinate their work and to make necessary decisions. Frequent meetings will be necessary, minimum once a week.

7 Utility Value of the Project and Deliverables

The utility value of this project is many-fold, namely:

1. One of the main objectives of this project is to train graduate students in VLSI physical design research, iterative algorithms and their parallelization aspect.
2. A new lab will be setup which will provide support for future research projects and help in research work of faculty and graduate students.
3. It is expected that the findings of our investigation of iterative search for near optimal solutions will help in addressing other NP-hard engineering

problems.

4. The results of this work can be used by other investigators in academia and industry to enhance existing methods.

8 Detailed Budget

Senior Investigators

The senior investigators will work for 24 months during the regular semesters for the entire duration of the project. They will receive payments as per the university regulations. Two graduate students will assist in the implementation aspects of the research. Two additional graduate students will be required in the preparation of setup, literature search, experimentation, etc. Their total compensation will be (14,400/- per Graduate Student/Research Assistant SR 57,600/-.)

Dr. Sadiq M. Sait (PI)	SR 1200/- * 24 =	SR 28,800
Dr. AbdulWaheed A.S. (CO-I)	SR 1000/- * 24 =	SR 24,000
Mr. Mahmood R. Minhas (CO-I)	SR 1000/- * 24 =	SR 24,000
Graduate Students	SR 600/- *4*24 =	SR 57,600
Total		SR 134,400/-

Equipment, Materials & Supplies, and Other Expenses

Facilities available at KFUPM will be used at no charge to the project. Additional equipment needed for the project includes 2xPentium 4 PC, 2.2GHz (or better), (Desktop/Portable), with a flat 19" TFT Monitor and Printer costing SR 30,000, networking equipments such as switches, hubs, etc will require SR 30,000/-, expenses for simple peripherals (such as CD writers, flash memories, hard-disks, remote keyboard and mouse, wireless devices, etc.,), consumables such as floppies, tapes, zip drives, CDs, printer toner, etc., will amount to SR 8,000/-, purchase of literature and books, stationary, etc., will require SR 3,000/-, photocopying charges, fax, telephones, etc., will amount to SR 1,000/-. Other miscellaneous and other incidental expenses may amount to a maximum of SR 1,200/-

Total cost of equipment, consumables and supplies is estimated to be

SR 38,200/-

A secretary will work for the entire duration of the project. SR 4,000/- for payments to secretary will be required. Individual items of the budget are summarized below.

Man Power	SR	134,400/-
Equipment:		
2 x Pentium 4, 2.2GHz (or better)(Desktop/Portable)		
with a 19" TFT Monitor & Printer.	SR	30,000/-
Networking equipments (switches, hubs, etc.)	SR	30,000/-
Consumables:		
Expenses for peripherals (such as CD writers, flash memories,		
hard-disks, remote keyboard and mouse, wireless devices, etc.,)		
Floppies, CDs, tapes, zip drives, printer toner, etc	SR	8,000/-
Books, other literature, Stationary, etc.,	SR	3,000/-
Photocopying charges, fax, telephones, etc.	SR	1,000/-
Secretary	SR	4,000/-
Miscellaneous and other incidental expenses	SR	1,200/-
Conference Attendance (2 Trips)	SR	30,000/-
Workshops/Short Course (2)	SR	30,000/-

The total cost [2] of the project is estimated to be SR 271,600/-

[2]SR 271,600/-= US $ 72,425/-

References

[1] Sadiq M. Sait and Habib Youssef. *VLSI Physical Design Automation: Theory and Practice*. McGraw-Hill Book Company, Europe, 1995.

[2] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms and their Application to Engineering*. IEEE Computer Society Press, December 1999.

[3] K. Shahookar and P. Mazumder. VLSI cell placement techniques. *ACM Computing Surveys*, 23(2):143–220, June 1991.

[4] Hirendun Vaishnav and Massoud Pedram. PCUBE: A performance driven placement algorithm for low power designs. *IEEE, Design Automation Conference, with Euro-VHDL*, 1993.

[5] A. Srinivasan, K. Chaudhary, and E. S. Kuh. Ritual: A performance-driven placement algorithm. *IEEE Transactions on Circuits and Systems - II*, 39(11):825–840, November 1992.

[6] S. Sutanthavibul, E. Shragowitz, and Rung-Bin Lin. An adaptive timing-driven placement for high performance VLSI's. *IEEE Transactions on Computer Aided Design*, 12(10):1488–1498, October 1993.

[7] Massoud Pedram. Logical-Physical Co-design for Deep Submicron Circuits: Challenges and Solutions. *Proceedings of Design Automation Conference, the ASP-DAC'98. Asia and South Pacific*, pages 137–142, February 1998.

[8] S. Kirkpatrick, Jr. C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, pages 498–516, May 1983.

[9] J. H. Holland. Adaptation in natural and artificial systems. *University of Michigan Press, Ann Arbor*, 1975.

[10] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., 1989.

[11] F. Glover, E. Taillard, and D. de Werra. A user's guide to tabu search. *Annals of Operations Research*, 41:3–28, 1993.

[12] R. Hübscher and F. Glover. Applying tabu search with influential diversification to multiprocessor scheduling. *Computers & Operations Research*, 21(8):877–884, 1994.

[13] F. Glover. Tabu search: A tutorial. *Technical Report, University of Colorado, Boulder*, February 1990.

[14] S. Nahar, S. Sahni, and E. Shragowitz. Experiments with Simulated Annealing. *Proceedings of 22nd Design Automation Conference*, pages 748–752, 1985.

[15] Prithviraj Banerjee. *Parallel Algorithms for VLSI Computer-Aided Design.* Prentice Hall International, 1994.

[16] M. Garey and D. Johnson. *Computer and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[17] A. Casotto, F. Romeo, and A. L. Sangiovanni-Vincentelli. A parallel simulated annealing algorithm for the placement of macro-cells. *IEEE Transactions on Computer-Aided Design*, CAD-6(5):838–847, September 1987.

[18] P. Banerjee and M. Jones. A parallel simulated annealing algorithm for standard-cell placement on a hypercube computer. *Proceedings of International Conference on Computer-Aided Design, ICCAD-86*, 1986.

[19] M. Toulouse, T.G. Crainic, and M. Gendreau. Issues in Designing Parallel and Distributed search Algorithms for Discrete Optimization Problems. *Publication CRT-96-36, Centre de recherche sur les transports, Université de Montréal, Montréal, Canada*, 1996.

[20] E. H. L. Aarts, Frans M. J. de Bont, Erik H. A. Habers, and Peter. J. N. Van Laarhoven. Parallel implementations of the statiscical cooling algorithm. *Integration, the VLSI Journal*, 4:209–238, 1986.

[21] Bernard Virot. *Parallelization of the Simulated Annealing Algorithm: Application to the Placement Problem.* Technical Report, University of Orleans, 1990.

[22] S. A. Kravitz and R. A. Rutenbar. Placement by simulated annealing of a multiprocessor. *IEEE Transactions on Computer-Aided Design*, CAD-6(4):534–549, July 1987.

[23] T. Starkweather, D. Whitley, and K. Mathias. Optimization using distributed genetic algorithm. In *Parallel problem solving from nature*, 1991.

[24] M. Tanese. Distributed genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 434–439. Morgan-Kaufmann, San Maeto, CA, 1989.

[25] M. Gorges-Schleuter. Explicit parallelism of genetic algorithms through population structures. In H. P. Schwefel and R. Männer, editors, *Problem Solveing from Nature*, pages 150–159. Springer Verlag, New York, 1991.

[26] N. Eldredge and S. J. Gould. *Punctuated Equilibria: An alternative to phyletic gradualism. Models of Paleobiology, T.J. M. Schopf, Ed. San Fransisco: CA, Freeman.* Cooper and Co., 1972.

[27] N. Eldredge. *Time Frames.* New York: Simon and Schuster, 1985.

[28] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. S. Ricichards. Punctuated equilibria: A parallel genetic algorithm. *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154, 1987.

[29] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. S. Ricichards. Distributed genetic algorithms for the floorplan design proboblem. *IEEE Transactions on Computer-Aided Design*, CAD-10:483–492, April 1991.

[30] V. Scott Gordon and Darrell Whitley. A machine-independent analysis of parallel genetic algorithms. *Complex Systems*, 8:181–214, 1994.

[31] M. Gorges-Schleuter. ASPARAGOS–An asynchronous parallel genetic optimization strategy. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms and their Applications*, pages 422–427. Morgan-Kaufmann, San Maeto, CA, 1989.

[32] I. De Falco, R. Del Balio, E. Tarantino, and R. Vaccaro. Improving search by incorporating evolution principles in parallel tabu search. In *Proc. of the first IEEE Conference on Evolutionary Computation- ICEC'94*, pages 823–828, June 1994.

[33] E. Taillard. Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 417:65–74, 1990.

[34] Bruno-Laurent Garica, Jean-Yves Potvin, and Jean-Marc Rousseau. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Computers & Operations Research*, 21(9):1025–1033, November 1994.

[35] T.G. Crainic, M. Toulouse, and M. Gendreau. Towards a taxonomy of parallel tabu search heuristics. *INFORMS Journal of Computing*, 9(1):61–72, 1997.

[36] S. Nair and A. Freville. A parallel tabu search algorithm for the 0-1 multidimentional knapsack problem. *11th International Parallel Processing Symposium*, April 1997.

[37] H. Mori and T. Hayashim. New parallel tabu search for voltage and reactive power control in power systems. *In Proc. of the 1998 IEEE International Symposium on Circuits and Systems - ISCAS'98*, pages 431–434, May 1998.

[38] R. M. Kling. *Optimization by Simulated Evolution and its Application to cell placement*. Ph.D. Thesis, University of Illinois, Urbana, 1990.

[39] R. M. Kling and Prithviraj Banerjee. Concurrent ESP: A placement algorithm for execution on distributed processors. *Proceedings of the IEEE International Conference on Computer-Aided Design*, pages 354–357, 1987.

[40] K. Ueda, T. Komatsubara, and T. Hosaka. A parallel processing approach for logic module placement. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 2(1):39–47, January 1983.

[41] A. Iosupovici, C. King, and M. Breuer. A module interchange placement machine. *Proceedings of 20th Design Automation Conference*, pages 171–174, 1983.

[42] Sadiq M. Sait and Habib Youssef. Timing-influenced general-cell genetic floorplanner. *Microelectronics Journal*, 28(2):151–166, 1997.

[43] Habib Youssef, Sadiq M. Sait, and K. Al-Farra. Timing-influenced force directed floorplanning. In *European Design Automation Conference Euro-DAC 95*, pages 156–161, September 1995.

[44] P. Cheung, C. Yeung, S. Tse, C. Yuen, and W. Ko. A new optimization cost model for VLSI standard cell placement. In *IEEE International Symposium on Circuits and Systems*, pages 1708–1711, June 1997.

[45] Sadiq M. Sait, H. Youssef, and Ali Hussain. Fuzzy simulated evolution algorithm for multiobjective optimization of VLSI placement. In *IEEE Congress on Evolutionary Computation*, pages 91–97, July 1999.

Resumé

Professor **Sadiq M. Sait**, Ph.D.

Work Address	**Home Address**
King Fahd University of Petroleum and Minerals	KFUPM Al-Nakheel Compound
KFUPM # 673	House Number 6526
DHAHRAN-31261	Dhahran-31261
Saudi Arabia	Saudi Arabia
(03)-860-2217 (Off)	Tel:(03)-860-6665 (Res)

Fax:(03)-860-3059 (Attn: Professor **Sadiq M. Sait**, COE Dept)

Family Status : Married with three children.
Date of Birth : October 28, 1957
Nationality : Saudi Arabian

Education

Ph.D. in ELECTRICAL ENGINEERING, (Computer Science Minor), King Fahd University of Petroleum and Minerals (KFUPM), November 1986, (GPA $\frac{4.0}{4.0}$). Thesis title: 'VLSI Mask Generation from Register Transfer Level Descriptions: An Automated Approach'.

M.S. in ELECTRICAL ENGINEERING, King Fahd University of Petroleum and Minerals (KFUPM), August 1983, (GPA $\frac{4.0}{4.0}$). Thesis title: 'Hardware Design and Software Development for a Microprocessor Based Application System'.

B.E. in ELECTRONICS ENGINEERING, Bangalore University, India, 1981, (Ranked 2^{nd} in the Entire University).

Employment History

Chairman, since 2000, Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Professor, since 1998, Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Served as Summer Coordinator, College of Computer Sciences & Engineering, June 2000–August 2000, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Associate Professor, 1992–1998, Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Summer Coordinator, July 1993–September 1993, Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Assistant Professor, 1987–1992, Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Lecturer, 1983–1986, Department of Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Research Assistant, 1981–1983, Department of Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

Academic Background

Computer Architecture, VLSI Design, Switching Theory, Micro-electronics, Hardware Description Languages, Logic Synthesis and Digital Design Automation.

Research Interest

VLSI Physical Design Automation, Microelectronics, High-Level Synthesis, Computer Architecture, Stochastic Search Algorithms.

Membership of Professional Societies

IEEE & IEEE Computer Society Saudi Arabian Computer Society.

Academic Honors

1. Invited **Keynote Speaker** at the **International Conference on Microelectronics**, ICM'95, Malaysia, December 1995.

2. Nominee for the **'Distinguished Teacher Award'** *three* times, and received the award *once* (from 'King Fahd University of Petroleum and Minerals'), in 1995.

3. Received the **'Distinguished Researcher Award'** *three* times from 'King Fahd University of Petroleum and Minerals', in 1990, 1994, and 1999.

4. Nominated by the Department of Computer Engineering for the **'Best Advisor Award'** *two* times.

5. **Editor** of *'Arabian Journal for Science and Engineering (AJSE)'* for Computer Science and Engineering 1992-present.

6. AJSE *'Arabian Journal for Science and Engineering'* Editor for the special issue on on Microelectronics.

7. Guest Editor for a special issue on 'Hardware Description Languages', for *International Journal of Computer-Aided VLSI Design,* 1989.

8. Member of the Editorial Board of *International Journal of Computer Aided VLSI Design,* 1987-1992.

9. Nominated to appear in **Marquis Who's Who in the World** (to be listed in 1996 Edition).

10. Awarded the **'Distinguished Teacher and Advisor Award'** (from 'King Fahd University of Petroleum and Minerals'), in 2000.

List of Publications of Professor Sadiq M. Sait

Publications in Refereed Journals

1. M. Masud and **Sadiq M. Sait**. 'Universal AHPL-A Language For VLSI Design Automation'. *IEEE Circuits and Devices Magazine,* September 1986, pp 8–14.

2. **Sadiq M. Sait** and M. A. Kulaib. 'A CMOS Cell for Parallelly Loadable Counters'. *International Journal of Electronics*, Vol. 62, No.6, 1987, pp 867–871.

3. A. A. Soomro, M. Rahman, and **Sadiq M. Sait**. 'A General Real-Time Decoder Based on AMD2900 Devices'. *Journal of Microprocessing and Microprogramming,* December 1987, pp 97–113.

4. A. A. Soomro, **Sadiq M. Sait** and M. Rahman. 'A Bit-Slice Micro-processor Based Decoder'. *Journal of Microprocessors and Microsystems,* December 1987, pp 527–534.

5. **Sadiq M. Sait**, A. Y. Yaagoub, and M. Masud. 'A CAD Tool for the Automatic Generation of Microprograms for Systems Modeled in UAHPL'. *Journal of Microprocessors and Microsystems,* October 1988, pp 463–470.

6. M. A. Kulaib, G. F. Beckhoff, and **Sadiq M. Sait**. 'Design of a Programmable Length Memory and its Controller'. *International Journal of Electronics*, Vol 65, No.2, November 1988, pp 923–932.

7. **Sadiq M. Sait** and F. A. Al-Khulaiwi. 'Automatic Weinberger Synthesis from a UAHPL Description'. *International Journal of Electronics*, Vol-69, No.2, 1990, pp 211–224.

8. **Sadiq M. Sait** and M. A. Al-Rashed. 'An Efficient Algorithm for Weinberger Array Folding'. *International Journal of Electronics*, Vol-69, No.4, 1990, pp 509–518.

9. **Sadiq M. Sait** and A. H. El-Maleh. 'A State Machine Synthesizer with Weinberger Arrays'. *International Journal of Electronics*, 1991, Vol 71, No.1, pp 1–12.

10. **Sadiq M. Sait**. 'Integrating UAHPL-DA System with VLSI Design Tools to Support VLSI DA Courses'. *IEEE Transactions on Education*, Vol 35, No.4, November 1992, pp 312–320.

11. **Sadiq M. Sait**. 'An Architecture to Store Path History in a Trellis and its Application to the Viterbi Algorithm'. *International Journal of Electronics*, 1992, Vol 72, No.1, pp 11–19.

12. J. Yazdani, M. Masud and **Sadiq M. Sait**. 'PCB Layout Generation from RTL Specifications'. *International Journal of Electronics*, 1992, Vol 72, No.1, pp 1–10.

13. **Sadiq M. Sait** and M. S. K. Tanvir. 'VLSI Layout Generation of a Programmable CRC Chip'. *IEEE Transactions on Consumer Electronics,* November 1993, Vol 39, No.4, pp 911–916.

14. M. S. T. Benten and **Sadiq M. Sait**. 'GAP: A Genetic Algorithm Approach to Optimize 2-bit Decoder PLAs'. *International Journal of Electronics*, 1994, Vol 76, No.1, pp 99–106.

15. M. S. T. Benten, **Sadiq M. Sait**, A. S. Al-Mulhem, and H. Youssef. 'RTL Structural Synthesis from Behavioral Descriptions in a Unix Environment'.

Arabian Journal for Science and Engineering, 19:4B, October 1994, pp 783–803.

16. **Sadiq M. Sait**, M. S. T. Benten, H. Youssef, and F. Soleja. 'Automated VHDL Composition from AHPL'. *Arabian Journal for Science and Engineering,* 19:4B, October 1994, pp 771–782.

17. M. S. T. Benten and **Sadiq M. Sait**. 'Genetic scheduling of task graphs'. *International Journal of Electronics*, 1994. Vol 77, No.4, pp 401–415.

18. **Sadiq M. Sait** and W. Hasan. 'Hardware Design and VLSI implementation of a Byte-Wise CRC Generator Chip'. *IEEE Transactions on Consumer Electronics,* Vol 41, No.1, February 1995, pp 195–200.

19. **Sadiq M. Sait**, M. S. T Benten, and A. M. T Khan. 'ASIC Design with UAHPL'. *IEEE Circuits and Devices Magazine,* March 1995, pp 14–24.

20. **Sadiq M. Sait** and A. A. Khalid. 'VLSI Design and Implementation of Systolic Queues'. *Journal of Microprocessors and Microsystems*, April 1995, Vol 19, No.3, pp 139–146.

21. H. M. Alnuweiri and **Sadiq M. Sait**. 'Efficient Network Folding Techniques for Routing Permutations in VLSI'. *IEEE Transactions on Very Large Scale Integrated (VLSI) Systems,* June 1995, pp 254–263.

22. **Sadiq M. Sait**, K. Elleithy and M. Hasan. 'Formal Synthesis of VLSI Layouts from Algorithmic Specifications'. *International Journal of Computer Systems: Science and Engineering*, UK, Vol 11, Number 2, March 1996, pp 67-81. 21 and 23).

23. H. Youssef, **Sadiq M. Sait**, A. S. Al-Mulhem, and M. S. T. Benten. 'High-level Synthesis from Purely Behavioral Descriptions'. *International Journal of Computer Systems: Science and Engineering*, UK, Vol 11, Number 5, 1996. September 1996, pp 125-139.

24. **Sadiq M. Sait**, S. Ali, and M. S. T Benten. 'Scheduling and Allocation in High-level Synthesis using Stochastic Techniques'. *Microelectronics Journal*, Elsevier Science Ltd, North Holland, Vol. 27, No. 8, October 1996, pp 693-712.

25. **Sadiq M. Sait** and H. Youssef. 'Timing Influenced General Cell Genetic Floorplanner', *Microelectronics Journal*, Elsevier Science Ltd, North Holland, Vol. 28, No. 8, March 1997, pp 151-166.

26. **Sadiq M. Sait**, A. A. Farooqui, G. F. Beckhoff. 'The Architecture of a Highly Reconfigurable RISC Data Flow Array (DF-RISC-A) Processor'. *International Journal of Electronics*, Vol. 83, No.4, August 1997. pp 493-518.

27. **Sadiq M. Sait** and Talal Maghrabi. 'Component Selection and Pipelining using Stochastic Evolution Algorithm'. (Manuscript Submitted) Journal of Computers & Electrical Engineering.

28. **Sadiq M. Sait** and Habib Youssef. 'CMOS/BiCMOS mixed design using Tabu Search'. IEE Electronics Letters, Vol. 34, No. 14, 1998, pp 1395-1396.

29. H. Youssef, **Sadiq M. Sait**, and K. Al-Farra. 'Timing Influenced Constraint Graph Based Force-Directed Floorplanning'. Manuscript **accepted** by *INTEGRATION, the VLSI Journal*, 1999. (Conference version already published, see Section , paper # 31).

30. **Sadiq M. Sait**, A. A. Farooqui, G. F. Beckhoff. 'A Novel Technique for Fast Multiplication', *International Journal of Electronics*, Vol 86, No.1, pp 67–77, January 1999.

31. **Sadiq M. Sait**, H. Youssef, K. W. Nassar, and M. S. T. Benten. Timing Driven Genetic Placement', *International Journal of Computer Systems: Science and Engineering*, Vol 14 No 1 January 1999, pp 3–14.

32. H. Youssef and **Sadiq M. Sait**. 'Timing Driven Global Routing for Standard Cell VLSI Design', *International Journal of Computer Systems: Science and Engineering*, Vol 14, No 3, May 1999, pp 175–186.

33. H. Youssef, **Sadiq M. Sait** and Hakim Adiche. 'Evolutionary Algorithms, Simulated Annealing, and Tabu Search: A Comparative Study,' *Engineering Applications of Artificial Intelligence*, IFAC, Pergamon, Vol 14, No 2, 2001. pp 167–181.

34. Habib Youssef, **Sadiq M. Sait**, E. Shragowitz, and H. Adiche. "Fuzzy Genetic Algorithm for Floorplanning", *Engineering Intelligent Systems*, CRL Publishing Ltd, UK, Vol 8, No. 3, September 2000, pp 145-153.

35. Habib Youssef, **Sadiq M. Sait**, and Ali Hussain. "Fuzzy Simulated Evolution Algorithm for VLSI Placement", accepted for publication in the *International Journal on Applied Intelligence*, Special issue on Applied Metaheuristics, Accepted, (to appear in June 2002).

36. Hasan Cam, Mostafa Abd-El-Barr, and **Sadiq M. Sait**. 'Design and Analysis of a High-Performance Hardware-Efficient Memory Allocation Technique'. The Journal of Systems and Software, June 2000 (Submitted).

37. **Sadiq M. Sait**, H. Youssef, H. Barada, and Ahmed Al-Yamani. "Parallellising Tabu Search on a Cluster of Heterogeneous Workstations", Journal of Heuristics, Special Issue on Parallel Metaheuristics, Vol 8, Number 3, May 2002.

38. H. Youssef, Abdulaziz Al-Mulhem, **Sadiq M. Sait**, M. Atif Khan. "QoS-Driven Multicast Tree Generation Using Tabu Search", The Computer Communications Journal on Advances in Performance Evaluation of Computer and Telecommunications Networking (Special Issue). Elsevier Science Ltd, May 2002 (to appear).

39. Salman A. Khan, Sadiq M. Sait, Habib Youssef. "Topology Design of Switched Enterprise Networks Using Fuzzy Simulated Evolution Algorithm", Engineering Applications of Artificial Intelligence Elsevier Science Ltd, March 2002 (Accepted).

40. Sadiq M. Sait and Munir M. Zahra. "Tabu Search Based Circuit Optimization", Engineering Applications of Artificial Intelligence Elsevier Science Ltd, February 2002 (Accepted).

Summary: Authored 40 journal papers, of these 38 have been accepted/published. I am the principal/single author in 23 of these, & second author in 15.

Book Chapters & Books (Authored and Edited)

1. **Sadiq M. Sait** and H. Youssef. Book Chapter entitled: *Modern Iterative Algorithms and their Applications in Computer Engineering*, in **The Computer Engineering Handbook**, December 2001 (accepted for inclusion), Editor Vojin Oklobdzija, CRC Press, Boca Raton, Florida, USA.

2. **Sadiq M. Sait** and H. Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems.* December 1999, IEEE Computer Society Press, California.

3. **Sadiq M. Sait** and H. Youssef. *VLSI Physical Design Automation: Theory and Practice*, McGraw-Hill Book Co., Europe, December 1994. Also Co-published by **IEEE Press**, USA, January 1995 (Hard bound edition).

4. M. S. T. Benten, **Sadiq M. Sait**, Abdul Raouf, and Samir H.Abdul-Jauwad. **Editors**. 'Advances in Microelectronics. Proceedings of the Fifth International Conference on Microelectronics', Dhahran. KFUPM Press, December 1993.

5. M. Masud and **Sadiq M. Sait** 'UAHPL Silicon Compiler—From RTL Specifications to Custom Layouts: Part-1: The Language and Its Compiler', *Progress In Computer Aided VLSI Design,* Ablex Publishing Corporation, New Jersey, 1990, pp 1–30.

6. **Sadiq M. Sait** and M. Masud. 'UAHPL Silicon Compiler—From RTL Specifications to Custom Layouts: Part-2: Automatic Generation of MOS Layouts', *Progress In Computer Aided VLSI Design,* Ablex Publishing Corporation, New Jersey, 1990, pp 31–64.

Publications in IEEE & International Refereed Conferences

1. **Sadiq M. Sait**. 'A General Cell Placement Procedure for UAHPL Based DA System'. *IEEE Proceedings of* **CompEuro'87,** Hamburg, May 1987, pp 513-514.
2. **Sadiq M. Sait** and M. Masud. 'CAD of Custom VLSI Layouts from RTL Specifications'. *30th Midwest Symposium on Circuits and Systems,* August 1987, Syracuse, New York, pp 554-558.
3. **Sadiq M. Sait**, M. Masud, and G. F. Beckhoff. 'Heuristics for Automatic Routing of Cells Placed by UAHPL Silicon Compiler'. *Second International Conference on Microelectronics and Microcomputers,* Menouf, Egypt, December 1987.
4. M. A. Kulaib, G. F. Beckhoff, and **Sadiq M. Sait**. 'CMOS Programmable Length First-In, First-Out Memory'. *Second International Conference on Micro-Electronics and Micro-computers,* Menouf, Egypt, December 1987.
5. **Sadiq M. Sait**, A. F. Damati, and M. Rahman. 'Systolic Architecture Design for Decoding Convolutional Codes using Viterbi Algorithm'. *Proceedings of International Conference on Mini and Microcomputers and Their Applications*, **MIMI'88**, Barcelona, Spain, June 1988, pp 526–529.
6. M. Masud, **Sadiq M. Sait**, and A. Y. Yaagoub. 'Automatic Generation of Microprograms for Systems Modeled in RTL'. *Proceedings of International Conference on Mini and Microcomputers and Their Applications,* **MIMI'88**, Barcelona, Spain, June 1988, pp 150–153.
7. M. Atiquzzaman and **Sadiq M. Sait**. 'A New Data Loading Technique in Multiprocessor Systems for Image Processing'. *Proceedings of International Conference on Mini and Microcomputers and Their Applications*, **MIMI'88**, Barcelona, Spain, June 1988, pp 457–460.
8. **Sadiq M. Sait**, A. F. Damati, and M. Rahman. 'A New Architecture for Viterbi Decoding and Its CMOS VLSI Implementation'. *31st Midwest Symposium on Circuits and Systems,* Missouri-Rolla, August 1988.

9. **Sadiq M. Sait** and M. Masud. 'Interfacing UAHPL DA System to Silicon Foundry'. *First International Conference on Micro-Electronics*, **ICM'88**, Algiers, November 1988.

10. **Sadiq M. Sait**, A. F. Damati, and M. Rahman. 'A Systolic Algorithm for VLSI Design of a $\frac{1}{n}$ Rate Viterbi Decoder'. *IEEE* **Melecon'89,** April 1989, Portugal.

11. M. Masud, J. Yazdani, and **Sadiq M. Sait**. 'Automatic Generation of PCB Layouts from Register Transfer Level Specifications'. (Accepted) *1989 International Symposium on Circuits and Systems.* China.

12. A. H. El-Maleh and **Sadiq M. Sait**. 'A State Machine Synthesizer with Weinberger Arrays'. *The IEEE Pacific RIM Conference,* Victoria, Canada, 1991.

13. H. Essam, **Sadiq M. Sait** and M. S. T. Benten. 'From Digital System Models in UAHPL to Layouts using ULMs'. (Accepted by) *First Great Lakes Symposium on VLSI*, **GLSVLSI'91**, Michigan, March 1991.

14. M. S. T. Benten and **Sadiq M. Sait**. 'Automatic Implementation of Data Link Controllers from High Level Language Descriptions of Protocols'. (Accepted by) *First Great Lakes Symposium on VLSI,* **GLSVLSI'91**, Michigan, March 1991.

15. A. M. T. Khan, **Sadiq M. Sait** and G. F. Beckhoff. 'VLSI Implementation of Controllers for Communication Protocols from their Petri Net Models'. *IEEE International Symposium on Circuits and Systems,* California, May, 1992.

16. A. M. T. Khan, **Sadiq M. Sait** and G. F. Beckhoff. 'High Level Synthesis of Controllers for Communication Protocols'. *Second Great Lakes Symposium on VLSI,* **GLSVLSI'92**, Kalamazoo, February, 1992, pp 114-121..

17. **Sadiq M. Sait**. 'UAHPL-DA System and VLSI Design Tools to Support VLSI DA Courses,' *SUNY Conference on Educational Technology*, SUNY College, Oneonta, May 27-28, 1992.

18. H. Al-Nuweiri, **Sadiq M. Sait** and M. Al-Darwish. 'Efficient Routing of a Class of Permutations in VLSI'. **BROWN/MIT** *Conference on Advanced Research in VLSI and Parallel Systems,* Providence, March 25-27, 1992.

19. **Sadiq M. Sait**, H. Youssef, F. Soleja, and M. S. T. Benten. 'Automated VHDL composition from AHPL'. *Fifth International Conference on Microelectronics*, **ICM'93**, December 1993, pp 220–224.

20. **Sadiq M. Sait**, M. S. T. Benten, and A. M. T Khan. 'ASIC Design from UAHPL Models'. *Fifth International Conference on Microelectronics*, **ICM'93**, December 1993, pp 237–241.

21. K. Elleithy, **Sadiq M. Sait** and M. Hasan. 'Formal Design of VLSI Systems'. *Fifth International Conference on Microelectronics*, **ICM'93**, December 1993, pp 214–219.

22. **Sadiq M. Sait**, M. S. T. Benten, and Asjad M. T. K. 'ASIC Design with AHPL'. *IEEE* **Melecon'94**, April 1994, pp 1234–1237.

23. **Sadiq M. Sait**, K. Elleithy, and M. Hasan. 'Design of a Cell Library for Formal High-level Synthesis', *IEEE* **Melecon'94**, April 1994, pp 1238–1241.

24. S. Ali, **Sadiq M. Sait**, and M. S. T. Benten. 'GSA: Scheduling and Allocation using Genetic Algorithm'. *European Design Automation Conference with Euro-VHDL*, **Euro-DAC'94**, Grenoble, September1994, pp 84-89.

25. S. Ali, **Sadiq M. Sait**, and M. S. T. Benten. 'Application of Tabu Search in High-level Synthesis of Digital Systems'. *International Conference on Electronics, Circuits and Systems,* **ICECS'94**, Cairo, December 1994, pp 423-428.

26. **Sadiq M. Sait**, A. S. Al-Mulhem, H. Youssef, and M. S. T. Benten. 'Hardware Specific Optimization in High-level Synthesis'. *International Conference on Electronics, Circuits and Systems*, **ICECS'94**, Cairo, December 1994. pp 418–422.

27. H. F. Al-Sukhni, H. Youssef, **Sadiq M. Sait**, and M. S. T. Benten. 'A New Loop Based Scheduling algorithm'. *IEEE Phoenix Conference on Computers and Communications*, **IPCCC**, March 1995, pp 76-81.

28. H. Youssef, **Sadiq M. Sait**, K. Nassar, and M. S. T. Benten. 'Performance Driven Standard-cell Placement Using the Genetic Algorithm'. *Fifth Great Lakes Symposium on VLSI*, **GLSVLSI'95**, Buffalo, USA, March 1995, pp 124-127.

29. **Sadiq M. Sait**, H. Youssef, K. Nassar, and M. S. T. Benten. 'Timing Driven Genetic Algorithm for Placement'. *IEEE Phoenix Conference on Computers and Communications*, **IPCCC**, March 1995, pp 403-409.

30. **Sadiq M. Sait**, A. A. Farooqui, G. F. Beckhoff. 'A Novel Technique for Fast Multiplication'. *IEEE Phoenix Conference on Computers and Communications*, **IPCCC**, March 1995, pp 109-114.

31. H. Youssef, **Sadiq M. Sait**, and K. Al-Farrah. 'Timing Influenced Force Directed Floorplanning'. *European Design Automation Conference with Euro-VHDL*, **Euro-DAC'95**, Brighton, September 1995, pp 156-161.

32. **Sadiq M. Sait**, H. Youssef, S. Tanvir and M. S. T. Benten. 'Timing Influenced General-Cell Genetic Floorplanner'. *Asia and South-Pacific Design Automation Conference,* **ASP-DAC'95**, Japan, September 1995.

33. G. F. Beckhoff, **Sadiq M. Sait**, and A. A. Farooqui. 'Highly reconfigurable RISC data flow array processor for DSP applications'. *The 6th International Conference of Signal Processing Applications & Technology*, **ICSPAT'95**, Boston, October 1995.

34. **Sadiq M. Sait**. 'Synthesis of digital systems in VLSI', (Invited Paper and Keynote Address). *The 7th International Conference of Microelectronics*, **ICM'95**, Kuala Lumpur, December 1995.

35. A. A. Farooqui, **Sadiq M. Sait**, and G. F. Beckhoff. 'Data Flow RISC Processor'. *The 2nd Australasian Conference on Parallel and Real-Time Systems,* **PART'95**, Australia, September 1995.

36. E. Shragowitz, H. Youssef, **Sadiq M. Sait**, and H. Adiche. 'Fuzzy Genetic Algorithm for Floorplan Design'. (Invited Paper, abstract submitted to) *International Conference on Applications of Soft Computing*, **SPIE'97**, 1997.

37. **Sadiq M. Sait** H. Youssefand Munir M. Zahra. 'Tabu Search Based Circuit Optimization'. (Accepted by) *Great Lakes Symposium on VLSI*, **GLSVLSI'98**, SW Louisiana, February 1998, pp 338-343.

38. Ta-Cheng, **Sadiq M. Sait** and W. R. Cyre. 'Performance and Interface Buffer Size Driven Behavioral Partitioning for Embedded Systems'. *9th International Workshop on Rapid Systems Prototyping*, **IEEE Computer Society Sponsored**, Leuven, Belgium, April 1998.

39. Ta-Cheng Lin, Sadiq M. Sait and W. R. Cyre. 'Buffer Size Driven Partitioning for HW/SW Co-Design'. IEEE International Conference on Computer Design, ICCD'98, Austin, USA, 1998.

40. Sadiq M. Sait, Habib Youssef and Ali Hussain. Fuzzy Simulated Evolution Algorithm for Multiobjective Optimization of VLSI Placement", IEEE Congress on Evolutionary Computation", July 1999, Washington DC, pp 91-97.

41. Hasan Cam, Mostafa Abd-El-Barr, and Sadiq M. Sait. 'A High-Performance Hardware-Efficient Memory Allocation Technique and Design'. IEEE International Conference on Computer Design, ICCD'99, Austin, USA, 1999, pp 274-276.

42. H. Youssef, **Sadiq M. Sait**, and Salman Khan. "Fuzzy Simulated Evolution Algorithm for Topology Design on Campus Networks", IEEE Congress on Evolutionary Computation", July 2000, San Diego, USA,

43. Hassan Barada, **Sadiq M. Sait**, and Naved Baig. "Task Matching and Scheduling in Heterogeneous Computing Environments using Iterative Heuristics", 13th International Conference on Parallel and Distributed Computing Systems, August 2000, Las Vegas, USA.

44. **Sadiq M. Sait**, H. Youssef, H. Barada, and Ahmed Al-Yamani. "A Parallel Tabu Search Algorithm for VLSI Standard-Cell Placement", IEEE International Symposium on Circuits and Systems", May 2000, Geneva,

45. H. Barada, **Sadiq M. Sait** and N. Baig. "Task Matching and Scheduling in Heterogeneous Systems Using Simulated Evolution", 10th Heterogeneous Computing Workshop in conjunction with IPDPS 2001, San Francisco, April 2001.

46. H. Youssef, **Sadiq M. Sait**, and Salman Khan. "Fuzzy Evolutionary Hybrid Metaheuristics for Network Topology Design", International Conference on Evolutionary Multi-Criterion Optimization, EMO'01, March 7-9, 2001, ETH Zurich, Switzerland (A Springer Publication). (Accepted).

47. Habib Youssef, **Sadiq M. Sait** and Ali Hussain. Adaptive Bias Simulated Evolution Algorithm for Placement", IEEE *2001 International Symposium on Circuits and Systems*, May 2001, Sydney, Australia, pages 355-358.

48. Junaid Khan, **Sadiq M. Sait**, and Salman Khan. A Fast Constructive Algorithm For Fixed Channel Assignment Problem", IEEE *2001 International Symposium on Circuits and Systems*, May 2001, Sydney, Australia, pages 65-68.

49. Aiman H. El-Maleh, **Sadiq M. Sait**, and Syed Z. Shazli. Test Pattern Generation. (Spector, L., E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors). 2001. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001. San Francisco, CA: Morgan Kaufmann Publishers. pages 1019-1025.

50. **Sadiq M. Sait**, Habib Youssef, and Junaid A. Khan. Fuzzy Evolutionary Algorithm for VLSI Placement, (Spector, L., E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors). 2001. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001. San Francisco, CA: Morgan Kaufmann Publishers. pages 1056-1063.

51. Junaid A. Khan, **Sadiq M. Sait**, and Abdulaziz S. Al-Mulhem. Algorithms for Channel Assignment Problem in Wireless Networks, SCI 2001, July 22-25, 2001, Orlando, Florida USA.

52. H. Youssef, **Sadiq M. Sait** and Salman Khan. An Evolutionary Algorithm for Network Topology Design. International Joint INNS-IEEE Conference on Neural Networks Washington DC, July 14-19, 2001.

53. Aiman Al-Maleh, **Sadiq M. Sait** and S. Z. Shazli. Evolutionary meta-heuristic for state justification in sequential ATPG. International Joint INNS-IEEE Conference on Neural Networks Washington DC, July 14-19, 2001.

54. **Sadiq M. Sait**, H. Youssef and Junaid Khan. Fuzzy Simulated Evolution for Power and Performance Optimization of VLSI Placement. International Joint INNS-IEEE Conference on Neural Networks Washington DC, July 14-19, 2001.

55. **Sadiq M. Sait**, H. Youssef Aiman El-Maleh and M. Minhas. Iterative Heuristics for Multiobjective VLSI Cell Placement. International Joint INNS-IEEE Conference on Neural Networks Washington DC, July 14-19, 2001.

56. H. Youssef, A. Almulhem, **Sadiq M. Sait**, and M. Atif Tahir. QoS-Driven Multicast Tree Generation Using Tabu Search. Proceedings of the 2001 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2001). Florida, July 2001.

57. **Sadiq M. Sait**, H. Youssef and Junaid Khan. Fuzzified Iterative Algorithms for Performance Driven Low Power VLSI Placement IEEE International Conference on Computer Design, ICCD'2001, Austin, September 23-26, 2001.

58. Ahmad Al-Yamani, Sadiq M. Sait, and Hassan R. Barada. "HPTS: Heterogeneous Parallel Tabu Search for VLSI Placement", IEEE Congress on Evolutionary Computation", May 2002, Honolulu, Hawaii, USA (Accepted).

59. Sadiq M. Sait, Mahmood R. Minhas, and Junaid A. Khan. "Performance and Low Power Driven VLSI Standard Cell Placement using Tabu Search", IEEE Congress on Evolutionary Computation", May 2002, Honolulu, Hawaii, USA (Accepted).

60. Junaid A. Khan and Sadiq M. Sait. "Fuzzy Aggregating Functions for Multiobjective VLSI Placement", IEEE International Conference on Fuzzy Systems", May 2002, Honolulu, Hawaii, USA (Accepted).

61. Junaid A. Khan, Sadiq M. Sait and Mahmood R. Minhas. "Fuzzy Biasless Simulated Evolution for Multiobjective VLSI Placement", IEEE Congress on Evolutionary Computation", May 2002, Honolulu, Hawaii, USA (Accepted).

62. M. Atif Tahir, H. Youssef, A. Almulhem, **Sadiq M. Sait**, Fuzzy based MultiObjective Multicast Routing Using Tabu Search, Proceedings of the 3rd International Conference on Internet Computing 2002, Las Vegas, June 2002.

Publications in Regional/National Refereed Conferences

1. **Sadiq M. Sait** and M. Masud. 'An Approach to Automate VLSI Design'. 10^{th} **NCC**, *National Computer Conference and Exhibition,* King Abdul Aziz University, Jeddah, February 1988, pp 190-205.

2. **Sadiq M. Sait** and M. Masud. 'Development of VLSI Design Facility at KFUPM'. 11^{th} **NCC** *National Computer Conference and Exhibition,* King Fahd University of Petroleum and Minerals, Dhahran, March 1989, pp 613–623.

3. M. Masud, J. Yazdani, and **Sadiq M. Sait**. 'PCB Layouts from RTL Descriptions: An Automated Approach'. 11^{th} **NCC** *National Computer Conference and Exhibition,* King Fahd University of Petroleum and Minerals, Dhahran, March 1989, pp 2–13.

4. F. A. Al-Khulaiwi and **Sadiq M. Sait**. 'Automatic Weinberger Synthesis from a UAHPL Description'. **GESS'90**, Gulf Expert Systems Symposium, Kuwait, May 1990, pp C70–C82.

5. M. A. Al-Rashed and **Sadiq M. Sait**. 'An Efficient Algorithm for Weinberger Array Folding'. **GESS'90**, Gulf Expert Systems Symposium, Kuwait, May 1990, pp C43–C57.

6. **Sadiq M. Sait**. 'From CLUs in UAHPL To Optimal Weinberger Arrays'. 12^{th} **NCC**, *National Computer Conference and Exhibition,* Riyadh, October 1990, 807–821.

7. **Sadiq M. Sait**, M. S. T. Benten, and H. Youssef. 'Modeling in VHDL for UAHPL Natives'. 13^{th} **NCC**, *National Computer Conference and Exhibition,* Riyadh, November 1992, pp 672-692.

8. **Sadiq M. Sait**, M. S. T. Benten, and A. M. T. Khan. 'VLSI CAD: The Saudi Arabian Experience'. 13^{th} **NCC**, *National Computer Conference and Exhibition,* Riyadh, November 1992, pp 611-623.

9. M. S. T. Benten, **Sadiq M. Sait**, A. Maasrani, and H. Youssef. 'RTL Structural Synthesis from Behavioral Descriptions in a Unix Environment'. 13^{th} **NCC**, *National Computer Conference and Exhibition,* Riyadh, November 1992, pp 279-299.

10. **Sadiq M. Sait**, K. Abdul Aziz and M. S. T. Benten. 'A Framework for the VLSI Implementation of a Systolic Tree Based Data Structures'. HCST, Applied Science University, Jordan, August 1994.

11. M. S. T. Benten, **Sadiq M. Sait**, A. M. T. Khan. *Load Balancing: A genetic approach*, Parallel and Distributed Computing, Kuwait, March 1995.

12. A. A. Farooqui, G. F. Beckhoff, and **Sadiq M. Sait**. 'Design, Modeling, and Implementation of a RISC Data Flow Array Processor'. Parallel and Distributed Computing, Kuwait, March 1995.

13. **Sadiq M. Sait** and Talal Maghrabi. 'Component Selection and Pipelining using Stochastic Evolution Algorithm'. 15^{th} **NCC** *National Computer Conference and Exhibition,* King Fahd University of Petroleum and Minerals, Dhahran, October 1997.

14. H. Youssef, **Sadiq M. Sait** and Osama Al-Haj Isa. 'Computer Aided Design of Structured Backbones'. 15^{th} **NCC** *National Computer Conference and Exhibition,* King Fahd University of Petroleum and Minerals, Dhahran, October 1997.

15. H. Youssef, S. M. Sait, and Salman Khan. "A Simulated Annealing Algorithm for Switched Campus Network Design", accepted for presentation at *18th National Computer Conference, NCC'2000, Dhahran, KSA*. November 2000.

16. H. Youssef, S. M. Sait, and Salman Khan. "Topology Design of Structured Campus Networks", in proceedings of the *7th IEEE Annual Technical Exchange meeting*, Dhahran, 18-19 April, 2000.

17. H. Youssef, Sadiq M. Sait, and M. Ahsan Siddiqui. "Data Flow Graph Allocation to Array Processors using Genetic Algorithm", accepted for presentation at *18th National Computer Conference, NCC'2000, Dhahran, KSA*. November 2000.

18. Salman A. Khan, Syed Z. Shazli, Junaid A. Khan, Sadiq M. Sait, " Distance Education and its Prospects in Saudi Arabia", In Proceedings of First Saudi Technical Conference and Exhibition, 18-22 Nov. 2000, Riyadh, Saudi Arabia, Vol. 1, pp 81-85.

Summary: Authored 80 conference papers, of these 61 in refereed IEEE/International conferences, and 18 in Regional/National Refereed Conferences.

ABDUL WAHEED
440 N. Winchester Blvd.# 79, Santa Clara, CA 95050
(408) 985-4895 (Home)
awaheed777@yahoo.com **(E-mail)**

PROFESSIONAL INTERESTS

Performance evaluation, computer networks, parallel and distributed systems, WWW servers, proxy caches, and multimedia systems

EDUCATION

Ph.D., Electrical Engineering, Michigan State University, East Lansing, MI (1997). GPA: 3.78/4.0
M.S., Electrical Engineering, Michigan State University, East Lansing, MI (1993). GPA: 3.84/4.0
B.Sc., Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan (1991)

EXPERIENCE

(8/01 – present)

Performance Engineering Consultant
Clients include:

- Streaming Networks
- Avaz Networks
- Associated Technologies, Inc.

Accomplishments:
- Product performance evaluation
- Network traffic characterization
- Deployment of caching and streaming media products
- Setting up facilities for network performance evaluation and cluster computing

(8/01 – present)

Assistant Professor, Computer Engineering Department, KFUPM, Dhahran, Saudi Arabia
Responsible for:

- Teaching graduate and undergraduate courses
- Research
- Advising graduate students

Accomplishments:
- Established a network performance measurement facility from scratch
- Compared web servers (Apache and IIS) and streaming servers (Darwin and Windows media)
- Evaluated campus network traffic using proxy logs at the edge of the network
- Designed and taught graduate courses on parallel architecture and performance evaluation with hands-on projects

(10/00 - 7/01)

Inktomi Corporation, Foster City, California
Position: Performance Software Engineer
Responsible for:

- Product performance evaluation
- Competitive performance measurements
- Experimental design and workload characterization
- Load testing of forward and reverse proxy servers
- Comparison of software and hardware based SSL engines

Accomplishments:
- Comparison of company's proxy servers with CacheFlow
- Comparison of reverse proxy servers with Windows Media Server, RealNetwork Server, and Darwin Streaming Server
- Comparison of reverse proxy servers with Apache server, Microsoft IIS, and CacheFlow server accelerator products
- Workload characterization of a leading ISP using proxy logs
- Development of profiling and automated testing and reporting tools

EXPERIENCE
(Cont.)
(7/00 - 9/00)

Trimedia Technologies, Inc., Milpitas, California
Position: Software Engineer
Responsible for:

- Host-target communication architecture	- Development of profiling tool

Accomplishments:
- Development of software infrastructure for communication with embedded target system in various configurations. Only an initial part of this project was finished that involved definition of a generic communication infrastructure and an implementation for a Linux host platform.

(5/97 - 7/00)

MRJ Technology Solutions/NASA Ames Research Center, Moffett Field, California
Position: Research Scientist
Responsible for:

- AIMS monitoring and visualization tools - Application parallelization for shared memory systems	- Measurement and trace-driven simulation based memory performance evaluation - Monitoring infrastructure for widely distributed computers (the grid)	- Performance comparison of LDAP servers

Accomplishments:
- Parallelization of CFD kernels using automated tools for high-end distributed shared memory systems
- Development of software tools for automated memory performance evaluation using measurements and compiler-generated memory access information
- Development of a monitoring infrastructure for fault-tolerant computing in widely distributed environments.

(8/92 - 5/97)

Michigan State University, East Lansing, Michigan
Position: Teaching and Research Assistant
Responsible for:

- Teaching of micro-controller and interfacing laboratories	- Research on visualization tools - Evaluation of instrumentation systems for parallel and distributed systems	- Tool development for distributed real-time systems

Background:
- Modeling, evaluation, and testing of the *Paradyn instrumentation system*
- Modeling, experiment design, simulation, evaluation, and development of software tool technologies for parallel, distributed, and real-time systems.
- Teaching undergraduate laboratory sections in the areas of *digital logic design*, *microprocessors and interfacing*, *instrumentation and measurement*, *circuit design*, and *fundamentals of electrical engineering*; helping the students with course material; helping the instructors with grading; and occasionally, substitute lecturing.

EXPERIENCE
(Cont.)
(5/94 - 8/94)

Hewlett-Packard Research Laboratories, Palo Alto, CA.
Position: Software Engineer (summer intern)
Responsible for:

- Distributed debugging environment	- Integration of debugging and performance visualization tools	- Integration in shared-nothing environment

Background:
- Use of multiple commercially available debuggers (HP, IBM, gdb, etc.)
- Use of multiple visualization tools (ParaGraph, Matlab, and AVS)
- Integration through X Windows based communication

HONORS

Thoman Fellowship, Michigan State University, August 1996. Selected for the Thoman Fellow Program based on *outstanding achievements in the Ph.D. program.*

Ph.D. Research Recognition, Computer Measurement Group (CMG), Dec. 1995 Ph.D. research proposal and initial results recognized by the *CMG* as a *valuable contribution to the field of computer performance evaluation and management*.

Summer Fellowship, Department of Electrical Engineering, Michigan State University, Summer 1993. Awarded a summer fellowship for *excellent standing in the graduate program.*

Bachelor's Degree with Honors, University of Engineering and Technology, Aug. 1991. **A**warded the B.Sc. degree with honors for *excellent overall performance in undergraduate courses*.

Merit Scholarship, University of Engineering and Technology, Jan. 1986–Aug. 1991. Awarded a scholarship throughout undergraduate studies for *excellent academic standings.*

PROFESSIONAL AND ANALYTICAL SKILLS

Performance Evaluation of Network Products

Experience of experiment design and performance measurements for web servers (Apache and IIS), proxy caches (Inktomi Traffic Server and CacheFlow Server Accelerators), streaming media servers (Real Networks, Windows Media, and QuickTime formats), and internet security products (software OpenSSL, nCypher, and CryptoSwift). Experienced with commonly used benchmarks and tools for above evaluations, including: Webstone, Webbench, Web Polygraph, CacheFlow Measurement Client, and SpamPro. Hands-on experience of setting up testbed for performance measurements of these network products.

Network Traffic Analysis

Analysis of real traffic at the network edges using log data obtained over long periods of time. Designed specialized tools and shell scripts for automatic analysis of these data to determine throughput, latency, and quality for different types of network transactions. In a number of cases, used these data to parameterize simulation models for detailed analysis of various systems under specific workload situations. This work involved using C/C++, awk, perl, matlab, opnet, and NS.

Instrumentation and Monitoring Systems

Skilled in developing *instrumentation systems* for collecting runtime information from scalable parallel, distributed, and real-time systems.

PROFESSIONAL AND ANALYTICAL SKILLS (Cont.)

Code Parallelization

Source to source transformations and use of automated tools for parallelizing sequential code for execution on distributed shared memory parallel systems.

Computer System Modeling

Experience in developing analytical models (using stochastic and operations analysis techniques) as well as simulation models (using C++) for evaluating the performance of computer systems.

Programming

Extensive programming experience in C and C++ *languages*, X Windows and Motif *libraries*, TCP/IP based Unix *IPC* techniques including BSD *sockets* and *RPC*; *multithreading*; and familiarity with Java and HTML.

- Tool Integration

Experience with developing technologies for integrating performance evaluation, visualization, debugging, and scheduling tools using an event-driven paradigm based on X Windows inter-client communication for HP Lab's *Vizir* and MSU's DARPA- sponsored *PG*^{RT} projects.

- Course Work

Graduate level course work in the areas of parallel and distributed systems, computer system performance evaluation, computer architecture, computer networks, stochastic processes, statistical pattern recognition, digital design using VHDL and CAD tools, FPGAs and custom computing machines, digital signal processing, and control systems.

THESES

Ph.D., May 1997

Instrumentation System Design, Modeling, and Evaluation
Major Areas: parallel and distributed computing, performance evaluation

M.S., Nov. 1993

Performance Data Modeling, Transformations, and Multiple-Domain Analysis Methods
Major Areas: parallel and distributed computing, performance visualization

BOOK CHAPTERS

Workload Characterization, edited by G. Kotsis, Austrian Computer Society, Oct. 1999 "Workload Characterization of CFD Applications Using Partial Differential Equation Solvers," with Jerry Yan.

State-of-the-Art in Performance Modeling and Simulation: Advanced Computer Systems, edited by K. Bagchi, J. Walrand, and G. Zobrist, Gordon and Breach Publishers, 1998
"Instrumentation Systems for Parallel Tools," with Diane T. Rover, pp. 35–54.

Debugging and Performance Tuning for Parallel Computer Systems, edited by M. L. Simmons, A. H. Hayes, D. A. Reed, and J. Brown, IEEE Computer Society Press, Dec. 1995
"Multiple Views of Parallel Application Execution," with Ming C. Hao, A. Karp, and M. Jazayeri, pp. 199–206.

REFEREED JOURNAL PUBLICATIONS

Cluster Computing, 4(1), April 2001
"An Evaluation of Alternative Designs for a Grid Information Service," with Warren Smith, David Meyers, and Jerry Yan, pp. 29– 37.

IEEE Transactions on Software Engineering, 24(6), June 1998

"Modeling and Evaluating Design Alternatives for an On-Line Instrumentation System: A Case Study," with Diane T. Rover and Jeffrey K. Hollingsworth, pp. 451– 470.

IEEE Concurrency, 6(2), April– June 1998

"Software Tools for Complex Distributed Systems: Toward Integrated Tool Environments," with Diane T. Rover, Matt W. Mutka, and Aleksandar Bakic, pp. 40– 54.

REFEREED CONFERENCE PUBLICATIONS (Performance Evaluation)

1. "An Evaluation of Alternative Designs for a Grid Information Service," Porceedings of the 9th IEEE International Symposium on High Performance Distributed Computing, with Warren Smith, David Meyers, and Jerry Yan, August 2000. Available on-line from: http://www.nas.nasa.gov/~wwsmith/papers/hpdc00-gis.pdf.

2. "Performance Modeling and Measurement of Parallelized Code for Distributed Shared Memory Multiprocessors," Proc. of the Sixth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '98), with Jerry C. Yan, *Montreal, Canada, July 19–24,*

1998.

3. "Workload Characterization of CFD Applications Using Partial Differential Equation Solvers," Proc. of Workshop on Workload Characterization in High-Performance Computing Environments, with J. Yan, *Montreal, Canada, July 19, 1998*.

4. "Modeling, Evaluation, and Adaptive Control of an Instrumentation System," Proc. of Real-Time Technology and Applications Symposium (RTAS '97), with D. Rover, M. Mutka, H. Smith, and A. Bakic, *Montreal, Canada, June 9–11, 1997*.

5. "Modeling, Evaluation, and Testing of Paradyn Instrumentation System," *Proc. of Supercomputing '96*, with Diane T. Rover and Jeffrey K. Hollingsworth. (CD- ROM/electronic proceedings only), *Pittsburgh, Pennsylvania, Nov. 17–22, 1996*.

6. "Performance Evaluation of an Integrated Instrumentation System," Proc. of Int. Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '96), pp. 267–271, with Diane T. Rover, *San Jose, California, Feb. 1-3, 1996*.

7. "A Structured Approach to Instrumentation System Development and Evaluation," *Proc. of Supercomputing '95*, with Diane T. Rover. (CD-ROM/electronic proceedings only), San Diego, California, Dec. 4-8, 1995.

8. "A Resource Occupancy Model for Evaluating Instrumentation System Overheads," *Proc. of the 20th Annual International Conf. of the Computer Measurement Group (CMG '95)*, pp. 1212–1223, with H.D. Hughes and D. T. Rover, *Nashville, Tennessee, Dec. 3–8, 1995*.

9. "A Schema for Specifying and Classifying Instrumentation Systems," Proc. of International Workshop on Computer Performance Measurement and Analysis, pp. 42–51, with Diane T. Rover, *Beppu, Japan, Aug. 20–23, 1995*.

10. "Instrumentation System Management in Concurrent Computer Systems," *Proc. of the Seventh SIAM Conf. on Parallel Processing for Scientific Computing*, with Vincent Melfi and Diane T. Rover, San Francisco, California, Feb. 15- 17, 1995.

11. "A Model for Instrumentation System Management in Concurrent Computer Systems," *Proc. of the Twenty-Eighth Hawaii Int. Conf. on System Sciences (HICSS '95)*, pp. 432–441, with Vincent Melfi and Diane T. Rover, *Maui, Hawaii, Jan. 3-6, 1995*.

12. "A Matrix Approach to Performance Data Modeling, Analysis and Visualization," *Proc. of MASCOTS '94*, pp. 137–141, with Bernd Kronmuller and Diane T. Rover, , *Durham, North Carolina, Jan. 31- Feb. 2, 1994*.

(Tool Development)

1. "Vista: A Framework for Instrumentation System Design for Multidisciplinary Application," *Proc. of MASCOTS '96, Tools Fair*, pp. 192–195, with Aleksandar Bakic, David Pierce, Matt. W. Mutka, and Diane T. Rover, San Jose, CA., Feb. 1-3, 1996.

2. "VIZIR: An Integrated Environment for Distributed Program Visualization," *Proc. of MASCOTS '95, Tools Fair*, pp. 288–292, with Ming C. Hao, Alan H. Karp, and Mehdi Jazayeri, Durham, North Carolina, Jan. 18- 20, 1995.

3. "A Toolkit for Advanced Performance Analysis," Proc. of MASCOTS '94, Tools Fair, pp. 376–380, with Bernd Kronmuller, Roomi Sinha, and Diane T. Rover, Durham, North Carolina, Jan. 31- Feb. 2, 1994.

4. "Performance Visualization of Parallel Programs," *Proc. of Visualization '93*, pp. 174–181, with Diane T. Rover. Case studies of the paper appeared in the video proceedings of *Visualization '93, San Jose, California, Oct. 25-29, 1993*.

5. "Multiple-Domain Analysis Methods," *Proc. of the Third ACM/ONR Workshop on Parallel and Distributed Debugging*, pp. 53–63, with Diane T. Rover. Proceedings appeared in *ACM SIGPLAN Notices*, 28(12), San Diego, California, May 17-18, 1993.

6. "Advanced Methods of Performance Data Processing and Analysis," *Proc. of the Seventh International Parallel Processing Symposium (IPPS '93)*, pp. 609–613, with Diane T. Rover and Markus Doetsch, Newport Beach, California, April 13-16, 1993.

(Parallelization of Sequential Code)

1. "A Comparison of Automatic Parallelization Tools/Compilers on the SGI Origin 2000," Proc. of High Performance Communication and Computing (SC '98), with M. Frumkin, M. Hribar, H. Jin, and J. Yan, Orlando, Florida., Nov. 7-13, 1998.

2. "Parallelization of NAS Benchmarks for Shared Memory Multiprocessors," Proc. of High Performance Computing and Networking (HPCN Europe '98), with Jerry C. Yan, *Amsterdam, The Netherlands, April 21–23, 1998.*

<u>INVITED PUBLICATIONS</u>

1. "Parallelization of NAS Benchmarks for Shared Memory Multiprocessors," with Jerry Yan, Future Generations of Computer Systems, Nov. 1999.

2. "Performance Measurement of Parallel and Distributed System Using On-Chip Counters," with Jerry Yan, The International Journal of Parallel and Distributed Systems and Networks, 1999.

3. "Modeling, Evaluation, and Adaptive Control of an Instrumentation System" with Diane T. Rover, Matt W. Mutka, Hugh Smith, Aleksandar Bakic, The International Journal of Parallel and Distributed Systems and Networks, 1999.

Mahmood-ur-Rehman Minhas

P.O .Box 491
KFUPM, Dhahran 31261, KSA.
Tel: 966-3-860 1490, Fax: 966-3- 860 3059
E-mail: minhas@ccse.kfupm.edu.sa

Summary of qualifications and research experience

I have Masters Degree in Computer Engineering as well as in Computer Science. I have been involved in a reasonable amount of research work in the areas of iterative algorithms and VLSI cell placement, under KFUPM-funded project and during my Masters thesis. I recently published three research papers in the area of iterative algorithms for VLSI cell placement in international refereed conferences. I have some practical experience of studying, implementing, and carrying out performance evaluation of various approaches for parallelization. My current research interests include parallelization of iterative algorithms, VLSI physical design automation, and multi-objective optimization.

Research Work / Projects

1. Worked on KFUPM research project titled "Iterative Heuristics for Timing and Low Power VLSI Standard Cell Placement". The project involved extensive research for designing and implementing iterative heuristics for optimisation of VLSI standard cell placement.
2. Worked on the study and comparison of the performance of two parallelization techniques namely multi-threading and Inter-Process Communication (IPC) under SOLARIS operating system. The performance of P-threads library was also studied.
3. Worked on the simulation and comparison of cache memory system and studied the sensitivity of cache memory to certain of its design parameters.
4. Developed a validation model for Network News Transfer Protocol (NNTP) using Promela modelling language and SPIN validator.

Publications

1. Sadiq M. Sait, Habib Youssef, Aiman El-Maleh, **Mahmood R. Minhas,** “Iterative Heuristics for Multiobjective VLSI Standard Cell Placement”, In proceedings of IEEE International Joint Conference on Neural Networks (IJCNN’2001), held in Washington, D. C., U.S.A., July 14-19, 2001.

2. Sadiq M. Sait, **Mahmood R. Minhas,** Junaid A. Khan, “Performance and Low Power Driven VLSI Standard Cell Placement using Tabu Search”, In proceedings of IEEE Congress on Evolutionary Computation (CEC’2002), held in Honolulu, Hawaii, U.S.A., May 12-17, 2002

3. Junaid A. Khan, Sadiq M. Sait, **Mahmood R. Minhas**, “Fuzzy Biasless Simulated Evolution for Multi-objective VLSI Placement”, In proceedings of IEEE Congress on Evolutionary Computation (CEC’2002), held in Honolulu, Hawaii, U.S.A., May 12-17, 2002.

Programming Assignments

1. Developed a compiler for a sub-set of Pascal.
2. Developed a personnel and payroll management system for Water and Power Development Authority (WAPDA), Islamabad, Pakistan, using C language
3. Developed a graphic package for automatic drawing of Entity-Relationship (ER) model used in designing of DBMS’s.
4. Developed a Sales Management System for Software Development Group (SDG), Ultimus Inc., Raleigh, NC, USA, using Visual Basic 5.0, MS Access and ODBC.

Job Experience

September 2001- present
Lecturer, Information and Computer Science Department, KFUPM, Dhahran, KSA

January 1999- September 2001
Research Assistant, Computer Engineering Department, KFUPM, Dhahran, KSA

1. Carrying out research in various fields of computer engineering.
2. Development of research-oriented simulators.
3. Assisting faculty members in research activities.

Academic Qualifications

M.S. Computer Engineering
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia - GPA: 3.50/4.00

Major Courses: Digital System Testing, Computer Networks, CAD of Digital Systems, Computer Architecture, Operating Systems, Design and Analysis of Computer Network Protocols. Computer System Performance Analysis, Applied Combinatorics and Graph Theory, Computer Network Programming, E-commerce.

M.S. Thesis: "Iterative Algorithms for Timing and Low Power Driven VLSI Standard Cell Placement”

M.Sc. Computer Science
International Islamic University, Islamabad, Pakistan - GPA: 3.38/4.00
Courses: Database Management Systems, Microprocessor and Assembly Language, Software Engineering, Compiler Construction, Operating Systems, Data Communication, Computer Networks, Computer Architecture, Programming Languages.

Short Course

Attended a short course on *“Modern digital system design using VHDL”* offered by Computer Engineering Department, from March 17 to March 21, 2001.

Awards and Honors

Received student travel award from IEEE, USA, for presenting my research paper in IJCNN held in Washington, D.C., in July 2001.

Membership

Student Member, IEEE.