ENHANCING SCHEDULING SOLUTIONS THROUGH ANT COLONY OPTIMIZATION

Shekhar Kopuri and Nazanin Mansouri

Syracuse University
Department of Electrical Engineering and Computer Science
123 Link Hall, Syracuse, NY, 13244-1240
(315) 443-2483 (Ph) (315) 443-1122 (Fax)
skopuri, namansou@ecs.syr.edu, http://web.ecs.syr.edu/∼namansou

ABSTRACT

In this paper a methodology based on Ant colony optimization is presented to generate optimal scheduling during high-level synthesis. The classical force equation of the Force-directed scheduling algorithm has been modified to accommodate the experiences accumulated by multiple agents in different iterations. In each iteration the obtained schedule is subjected to remaining steps of synthesis using standard techniques like clique partitioning for resource allocation and left edge algorithm. The results are used to improve scheduling in the next iteration.

1. INTRODUCTION

Ant-colony optimization(ACO) technique is a robust multi-agent population based approach to explore the solution space of a multi constraint optimization problem. First proposed by Dorigo et al in 1992, this technique is inspired by the ability of real ants to find the shortest path from the nest to the food source, collectively. ACO has been successfully applied to many scientific problems, especially those that are NP hard in nature. In this work, an Ants-based approach to solve the scheduling problem during high-level synthesis has been presented.

High-level synthesis can be defined as automated transformation of the behavioral representation of a hardware system into a structural design at the register transfer level (RTL). The design is synthesized using a library of components such that it conforms to a set of pre-defined constraints. The entire synthesis problem consists of many NP-complete multi-constraint optimization subtasks. The data-path and the controller are generated separately. The data-path synthesis is broadly divided into (1) *Scheduling*, the process of identifying individual operations, and assigning a time-step to each of these such that no dependencies are violated, (2) *Resource Allocation and Binding*, the process of identifying possible functional units that can perform each operation, and binding the operation to particular instances of these (3) *Register Allocation and Binding*, or the process of identifying the number and (optimized) assignment of various registers to store the result of each operation and (4) *Steering logic generation* or the process of generating the steering logic for the entire data-path (that can be performed differently depending upon whether the architecture has point to point, i.e. multiplexer-based or Bus-based interconnects).

This paper presents an ACO-based technique that generates a valid scheduling and optimizes it iteratively. The scheduling approach is inspired by the famous Force Directed Scheduling (FDS) algorithm proposed by Paulin and Knight in 1987 [1]. The force equation of the FDS has been modified by replacing the successor-predecessor force (PS-force) term with two new terms *viz. Trail* and *Global Experience*. These terms characterize the experience of one agent in generating a solution and the experience of the entire population, respectively. In each iteration a schedule is obtained which is subjected to the complete synthesis process. The results of the synthesis are used to update the experience terms in the next iteration. This iterative process yields a more realistic final schedule.

The remaining sections of this paper discuss this process in detail and are organized as follows. Section 2 is a review of the related research in ACO and scheduling in high-level synthesis. Section 3 presents the proposed methodology. The results are presented in Section 4. Finally, Section 5 draws conclusions and discusses the avenues for future work.

2. RELATED WORK

Ant colony optimization was first introduced by Dorigo et al [2, 3] to address combinatorial optimization problems. Since then it has been successfully applied to many multi-constraint practical problems[4, 5, 6, 7]. Dorigo et al proposed a multi-agent auto-catalytic methodology to solve multi-

This work is partly supported by a grant from MDC-NYSTAR, under award number 411903-G.

constraint optimization problems inspired by the ability of real ants to find the shortest path from nest to food source following the trials of a chemical substance called pheremone laid by other ants. The ants collectively strengthen the best trail over time while other less likely trails are weakened as pheremone is volatile. The first ants-based algorithm proposed was the AS algorithm(Ant System). Since then many modifications and versions suitable to specific problems have been proposed but all share the common features of (1) being a multi-agent population based optimization technique, (2) having a pheremone trail model that exhibits stigmergy (i.e. diminishing trial), and (3) taking decisions to change states probabilistically.

This paper attempts to solve the scheduling problem of high level synthesis using an Ant-based approach. Most scheduling algorithms used in High-level synthesis are derivatives of the Force-directed Scheduling algorithm presented by Paulin and Knight in 1987 [1]. Park and Kyung [8] proposed an iterative technique based on the graph-bisection problem by Kernighan and Lin. More recent research attempts combine different steps of synthesis: Heijeligers et al [9] and InSyn [10] use techniques like genetic algorithms and simulated evolution. One of the first attempts to use Ant-optimizations in High level synthesis is the work by Kienprasit and Chongstitvatana [11]. They proposed a system to do synthesis using dynamic ants. Their method uses a decision path graph which is initialy generated for each ant. Dynamic niche sharing is used to find the peaks which are updated within each cycle. This is iteratively improved over a population. They also suggested some heuristics that were generated faster results.

This paper presents an algorithm that tries to solve, specifically, the scheduling step of a high level synthesis process. Significantly, the algorithm has many parameters that can be determined empirically to make it adaptable to a typical synthesis framework. In other words, the methodology is less dependent on the type of architectures that are synthesized. It uses a modified force equation of the FDS algorithm. The experience of the entire population is based on the results of the synthesis process. The following section presents the the proposed algorithm.

3. METHODOLOGY

Force Directed Scheduling (FDS) algorithm has been widely used by many researchers in various forms since it was first proposed by Paulin and Knight [1]. It tries to minimize the overall concurrency under a fixed latency. At every timestep, the effect of scheduling an unscheduled operation is calculated, and the one with least worse effect is selected. This effect is termed as force, and comprises of two components: the self-force, S_{il}, and the predecessor-successor (PS) forces, PS_{il}, of all its predecessors and successors. Let an unscheduled operation i be scheduled at a time step l. Then, the self-force S_{il} represents the direct effect of this scheduling on the overall concurrency. It is given by:

$$S_{il} = \sum_{m=t_i^S}^{t_i^L} q_k(m)(\delta_{lm} - p_i(m)) \quad (1)$$

where, t_i^S and t_i^L are the ASAP and ALAP times respectively, k is the type of operation, q_k is the type distribution for type k and δ_{lm} is the Kronecker delta function. This scheduling might cause the time frame of a predecessor or successor operation to change from $[t_i^S, t_i^L]$ to $[\tilde{t}_i^S, \tilde{t}_i^L]$. Then the ps-force exerted by this predecessor or successor is given by

$$PS_{il} = \frac{1}{\tilde{\mu}_i + 1} \sum_{m=\tilde{t}_i^S}^{\tilde{t}_i^L} q_k(m) - \frac{1}{\mu_i + 1} \sum_{m=t_i^S}^{t_i^L} q_k(m) \quad (2)$$

where μ_i and $\tilde{\mu}_i$ are the mobility associated with the original and the changed time frames.

The FDS algorithm is of the order of O(n^3) time complexity. The results obtained are usually optimal but the method is time consuming, making it unsuitable for large graphs. Also, the basic FDS algorithm tries to reduce the number of functional units and memory units used. However, in the DSM realm the cost of the resources might not necessarily be dominant. The ACO-based technique presented here, that solves the scheduling problem for a *generic* circuit, has a modified force equation with the PS-force term replaced by two new terms the *Trail*, M_{il}^n and the *Global Experience*, G_{il}, representing the experience of one agent and that of the entire population, respectively. These experience terms are calculated based on the effectiveness of the schedule after complete synthesis. The process is captured algorithmically in the algorithm *Ant_sched*,

---

$G_{il} = 0; i \in [1, N],\ \ l \in [1, \lambda]$
$s^o = 0$; No best schedule
for n=1 to $n\_iterations$ **do**
  $M_{ij}^n = 0; i \in [1, N],\ \ l \in [1, \lambda]$
  $n_{ants} = |\eta N|; \eta < 1$
  calculate $S_{ij}\ \ i \in [1, N],\ \ l \in [1, \lambda]$
  sort S in descending order;
  initialize n_{ants} with the least n_{ants} self-forces
  for m = 1 to n_{ants} **do**
    s = procedure $A\_FDS()$;
    calculate $cost(s)$;
    update M_{ij}^n using equation 3
    if $cost(s^o) > cost(s)$
      $s^o = s$;
  update G_{ij} using equation 4

Table 1. Algorithm Ant_sched

n = 0;
while $n \neq N$;
compute S_{ij} $i \in [1, N]$, $l \in [1, \lambda]$
compute F_{ij} $i \in [1, N], l \in [1, \lambda]$ using equation 5
choose an operation and time-step at random amongst
x of those with least forces: $x = |\eta n|, \eta < 1$
schedule the operation and update the time frames
n = n +1;

Table 2. Procedure A_FDS

and the procedure *A_FDS*. The subsections 3.1 and 3.2 interpret these algorithms elaborately. It is important to note that the self-force can be calculated in linear time where as the the time complexity of the ps-force is at least O(n^2)[1].

3.1. ACO-Based Scheduling Algorithm

Table 1 presents the ants-based scheduling algorithm. The experience of the ants within n-th iteration is maintained in a matrix $M^n(N, \lambda)$, where N is the number of operations to be scheduled and λ is the latency found by ASAP. Any element $M^n_{i,l}$, therefore, is the trail deposited by the n-th ant and represents their experience of scheduling operation i at the time-step l (i.e. the experience in the n-th iteration). It is initialized to a matrix of zeros. The initial self forces are calculated and sorted. A fixed fraction η of the N operations $s.t.(n_{ants} = |\eta N|)$ with the least self-forces are selected, and each is assigned to an ant that is initialized with corresponding operation and time-step as the first scheduled operation. Each ant then performs the rest of the scheduling process using the procedure A_FDS described in Table 2. The resulting schedule, s, is used in subsequent synthesis steps, and the process runs to completion. The cost of the synthesized design is calculated using the costs described in Table 3. It may be noted that the mechanism to calculate costs can be varied according to the type of circuits one is dealing with. Therefore, the effectiveness of the algorithm is dependent on how realistic is the cost calculation. Then, the trail matrix for the n-th cycle, M^n, is updated as follows:

$$M^n_{ij} = M^n_{ij} + \frac{cost(s) - cost(s^o)}{cost(s^o)} \ \forall (i, j) \in s \quad (3)$$

where, s^o is the best schedule thus far. If the schedule results in an improved design, s^o is changed to s. The trail laid by an ant is positive or negative depending on whether the improved synthesis results are obtained or not. The strength of the trail is proportional to the degree of improvement or worsening of the design. The trail evaporates by a factor ρ. A global matrix,$G(N, \lambda)$, to collect the trails laid by the entire population in all the cycles is maintained and is updated after every cycle in the following way

$$\forall (i, j) \quad G^{n+1}_{ij} = \rho G^n_{ij} + M^n_{ij} \quad (4)$$

3.2. The Force Equation

The A_FDS uses following force equation to calculate force exerted when an operation i is scheduled at time-step l

$$F_{il} = \alpha \frac{S_{il}}{\sqrt{\frac{1}{N\lambda} \sum_{(i,j)=(1,1)}^{N,\lambda} S_{ij}}} - \beta M^n_{il} - \gamma G^n_{il} \quad (5)$$

The first term is the normalized self-force component. The second term captures the experience accumulated in the current cycle and the third term captures the experiences accumulated over all the cycles. It should be noted that with increasing cycles the last two components increase in value, hence diminishing the importance of the the self-force over time. This is important as the self-force does not take the forces exerted by other operations into account. However, it can give a good starting point for the iterative process and eventually the algorithm $learns$ this with experience. α, β, γ are constants determined by empirical means.

4. RESULTS

The algorithm Ant_sched was applied to four synthesis benchmarks viz. the Differential Equation, the Elliptical filter and two Discrete Cosine Transform benchmarks. Iterative improvement was seen in all the cases. Shown here in fig 1, fig 2 and fig 3 are the later three.Various values for the parameters $\eta, \lambda, \alpha, \beta, \gamma$ and ρ were simulated. Optimal schedules that led to minimal costs after synthesis were found in all the three cases within 20 iterations. Number of ants per iteration were 20. The values of η, α, β and γ were 0.5, 1, 2 and 2 respectively. ρ between 0.35 and 0.5 led to better solutions. As compared to the standard FDS algorithm the proposed algorithm generated better results (Table 4).

All the simulations were run on a Pentium 3, 800 MHz machine with 320Mb of RAM. None of the cases took more than few seconds to generate results. It is worthwhile to note that all the terms in the modified force equation can be calculated in linear time. The value of $n\_iterations$ is also low (<20 for the benchmarks). Also, the methodology is not bound by any specific allocation and binding technique nor limited to any specific cost estimation mechanism. It can be successfully implemented by varying the various parameters.

5. CONCLUSION

In this paper an ACO-based methodology to solve the scheduling problem in high-level synthesis for a generic design has

Hardware Unit	Cost
Single Cycle Multiplier	250
Single Cycle Adder/Subtractor/Comparator	50
Register	15
2-input Mux	15

Table 3. Cost Table

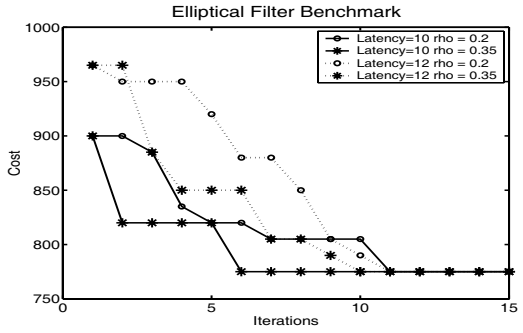

Fig. 1. Ant_sched on Elliptical filter Benchmark

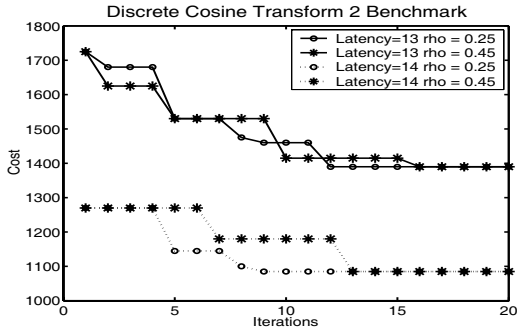

Fig. 3. Ant_sched on DCT2 Benchmark

been presented. Though iterative synthesis seems a time consuming step, we claim that this paper presents a new perspective to generating solutions to various individual steps of high level synthesis, each of which can be seen as a multi constraint optimization problem. Our method is independent of the architectures of the designs and tends to learn using the experience of a population instead of an individual. Another significant aspect is that we have used the classic FDS algorithm and modified it in a very simple form and generated solutions that developed iteratively. Our efforts in the future are targeted at modeling each of the individual problems of high level synthesis using Ant based optimization techniques and to perform extensive simulations to generate realistic values for the parameters in the algorithms. We also aim to tie them together to reduce the overall synthesis time.

6. REFERENCES

[1] John P. Knight Pierre G. Paulin, "Force-directed scheduling in automatic data path synthesis," *DAC*, pp. 195–202, 1987.

[2] M. Dorigo, *Optimization, Learning and Natural Algorithms*, PhD Thesis, Dipartimento di Elettronica, Politecnico di Milano, IT, 1992.

[3] M. Dorigo, V. Maneizzo, and A. Colorni, *Positive Feedback as a Search Strategy*, Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, IT, 1991.

[4] B.Bullnheimer, C.L. Mallows, and I.A. Wagner, *An Improved Ant System for Vehicle Routing Problem*, Technical Report POM-10/97 ,Institute of Management Science, University of Vienna, 1997.

[5] M. Dorigo, V. Maneizzo, and M. Trubian, "Ant system for job scheduling," *Belgian Journal of Operations Research, Statistics and Computer Science*, vol. 34, pp. 39–53, 1994.

[6] G.M. Gambardella and M. Dorigo, *HAS-SOP: A Hybrid AntSystem for Sequential Ordering Problem*, Technical Report 11-97, IDSIA, Lugano, CH, 1997.

[7] D. Costa and A. Hertz, "Ants can color graphs," *Journal of Operational Research Society*, vol. 48, pp. 295–305, 1997.

[8] I.C. Park and C.M. Kyung, "Fast and near optimal scehduling for data path synthesis," *DAC Proceedings*, pp. 650–685, June 1991.

[9] M.G.M. Heijligers, L.J.M. Cluitmans, and J.A.G. Jess, "High level sythesis, scheduling and allocation using genetic algorithms," *ASP-DAC*, pp. 61–66, 1995.

[10] A. Sharma and R.Jain, "Insyn : Integrated scheduling for dsp applications," *International Synposium on High level Sythesis*, pp. 96–103, 1994.

[11] R. Keinprasit and P. Chongstitvatana, "High level synthesis by dynamic ants," *International Journal of Intelligent systems*, 2003.

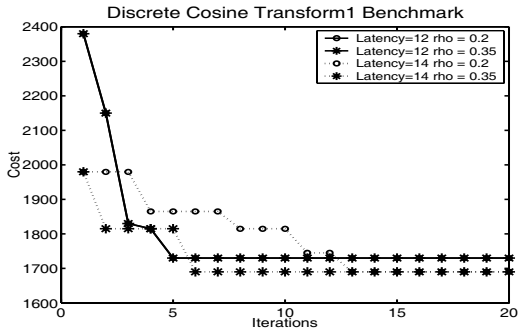

Fig. 2. Ant_sched on DCT1 Benchmark

Benchmark	Latency	FDS	A_FDS
Differential equation	4	875	875
	5	875	875
Elliptical filter	10	825	775
	12	775	775
Discrete Cosine Transform 1	12	1795	1730
	14	1795	1690
Discrete Cosine Transform 2	13	1210	1085
	14	1525	1390

Table 4. Cost of best solution for FDS and A_FDS procedures