Review of: Application Partitioning on Programmable Platforms using the ACO

1. The authors refer to previous work done on codesign problems using ACO. The codesign problem is not that much different from the partitioning problem. Yet, the authors provide no description of that previous ACO codesign work in this paper.

The authors should describe that previous ACO codesign work here. Be sure to include a clear explanation about why it was unsuited for the partitioning problems considered in this paper. That will establish the need for a new ACO method.

But this raises another issue. The authors claim the codesign problem is just a special case of the partitioning problem---which means the partitioning ACO method described in this paper should work on codesign problems as well. It would therefore seem reasonable to compare the two ACO methods on the same codesign problems. Why was this was not done?

2. Section IV needs far, far more detail. It is not at all clear how a given task execution time or area requirement was determined. The authors just give vague and ambiguous statements. For example, they claim "software code length is estimated based on the number of instructions needed to encode the operations of the CDFG." So, how were the number of instructions needed for the CDFG encoding determined? How was this number converted into bytes of memory? I also don't understand how ASAP scheduling translates into the number of CLBs used in the FPGA (which the authors claim can be done). With the level of detail provided, it is impossible for anyone to duplicate what the authors have done.

3. The statistical analysis it too long and of little value. The fact that the ACO found good solutions most of the time is frankly unconvincing. This may be nothing more than a consequence of the nature of the problems chosen for testing. I'm not sure how these statistics can be extended to cover task graphs with different constructs. (See related remarks in items 5.)

4. I also was not convinced by the comparisons with simulated annealing. It is unfair to compare an ACO run against a single SA run regardless of how long the SA actually ran. Actually I was confused about something. Did the authors compare their ACO against a single SA run or against the best of a number of SA runs where each run was initialized differently? Please clarify.

5. In my opinion the test cases chosen for study are too simple to make any real conclusions about their proposed ACO method. For example, the authors ignored task graphs with loops. This greatly simplifies task scheduling because only a topological sort is needed . (As a side question, do cyclic subgraphs pose any special problems for ACO methods?) The authors did not consider any task temporal constraints such as deadlines or latest start times even though real-time constraints determine a partition's feasibility in many systems. The authors assumed all task execution times were static. Yet, both execution time and power consumption can be data dependent. It is not uncommon to find partitioning problems formulated as multiobjective problems so that tradeoffs between say schedule length and power consumption can be investigated; the authors only considered a single objective problem.

To summarize, the authors have avoided all of the really nasty design issues that just happen to be of real interest. I therefore doubt their ACO method will generate much interest in the hardware/software codesign community. More extensive and realistic testing is needed before making any conclusions about the ACO method efficacy.

After considering all of the above problems, I cannot recommend publishing this paper.