Optimized Simulated Annealing for Network-on-Chip Application Mapping

Ciprian Radu, Lucian Vinţan

Advanced Computer Architecture & Processing Systems Research Lab
"Lucian Blaga" University of Sibiu, Romania
(e-mail: {ciprian.radu, lucian.vintan}@ulbsibiu.ro)

Abstract: Network-on-Chip (NoC) application mapping is an NP-hard problem, which can be addressed with heuristic algorithms. This paper presents an Optimized Simulated Annealing (OSA) algorithm that deals with the topological placement of cores onto NoC nodes. The algorithm is derived from a general energy- and performance-aware Simulated Annealing and employs and adapts some of the best Simulated Annealing practices from the field of task scheduling. OSA uses an application- and network-based exploration of the search space. The cores are implicitly and dynamically clustered using knowledge about communication demands. We show that OSA is a more feasible Simulated Annealing approach to NoC application mapping by comparing it with a general Simulated Annealing algorithm and a Branch and Bound algorithm, too. Our simulations use real applications and show OSA to be more than 98% faster than a general Simulated Annealing, without giving worse solutions. Also, compared to a Branch and Bound technique, it gives better solutions, as the problem size increases, while in terms of speed and memory consumption the two algorithms are comparable.

Keywords: Network-on-Chip (NoC), application mapping, clustering, optimization, evaluation, simulation

1. INTRODUCTION

Simulated Annealing (SA) is a stochastic heuristic search technique introduced by Kirkpatrick et al. (1983) as a generalization of the Metropolis method, which simulates a system of particles that suffers changes in temperature. Kirkpatrick et al. used Simulated Annealing to attack a classical NP-hard optimization problem, the traveling salesman problem, and they also applied SA to NP-hard computer design problems like partitioning, component placement and wiring of electronic systems.

The advantages of Simulated Annealing are its ease of implementation, its applicability for many combinatorial optimization problems and the ability to give reasonably good solutions. However, the parameters of the algorithm must be carefully chosen, since SA can easily run for a very long time until it gives a suitable solution. Because Simulated Annealing is a very general algorithm, several generic and problem-specific choices have to be made in order to implement it for a particular problem Fouskakis and Draper (2007).

Simulated Annealing is used in the field of Network-on-Chip (NoC) application mapping. The NoC application mapping problem is the topological placement of the Intellectual Property (IP) cores onto the on-chip tiles Hu and Marculescu (2003). This is an NP-hard problem because the number of possible mappings is ${}_nP_k = \frac{n!}{(n-k)!}$. Therefore, heuristic algorithms are proposed to solve this problem.

We propose an Optimized Simulated Annealing (OSA) technique for the NoC application mapping problem. We begin by presenting some related work. After we describe OSA, we briefly present our simulation methodology. Next, we present OSA's performance in terms of speed, memory consumption and solution quality, by comparing it with a general Simulated Annealing and a Branch and Bound (BB) algorithm. Finally, we present our conclusions and directions for future work.

2. RELATED WORK

Simulated Annealing was one of the first algorithms used to address the NoC application mapping problem Hu and Marculescu (2003). Using a simple energy analytical model, the authors proposed an energy-aware mapping for square, $N \times N$, 2D mesh NoCs, with XY routing and wormhole switching. Two algorithms were used to heuristically solve the problem: Simulated Annealing and Branch and Bound. Both of them are bandwidth constrained and try to find the best solution by minimizing the communication energy. Hu and Marculescu showed that SA is capable of finding better mappings than the ones found using Branch and Bound. SA's main disadvantage is speed: the simulation results from Hu and Marculescu (2003) show that BB is tens of times faster than SA. For a 10x10 NoC, SA did not finish its execution in 40 hours. Both SA and BB algorithms were further developed in Hu and Marculescu (2005), where the mapping problem is treated in conjunction with the routing problem.

Little research has been done to optimize Simulated Annealing for mapping cores onto NoC tiles. A Cluster-based

Simulated Annealing (CSA) was proposed by Lu et al. (2008). CSA tries to exploit the application's communication locality so that it can identify clusters of IP cores. Firstly, the NoC is clustered: the nodes are grouped based on the distance between them and their connectivity. Secondly, cores are clustered based on communication. A core cluster is then associated with each NoC node cluster. At high temperatures, the annealing process occurs normally: any cores may be moved. However, when the temperature decreases enough, the annealing process is limited to the cores from clusters. The clustering technique is static and it is driven first by the NoC topology and second by the application. Clustering represents a problem-specific choice for Simulated Annealing. It improves SA's speed because it uses more knowledge about the NoC application mapping problem than a general Simulated Annealing.

We propose next an Optimized Simulated Annealing algorithm for NoC application mapping. As compared to CSA, our algorithm performs an implicit clustering, during the annealing process. As we will show next, it is also significantly faster than both SA and CSA because OSA's search space exploration is application and network dependent. We do not cluster the NoC nodes because we do not want to restrict the core clustering process. Therefore, we apply clustering only for the application, and we do not explicitly cluster the IP cores. We rather influence the moves during the annealing process, so that the IP cores that communicate with each other clump together.

3. OPTIMIZED SIMULATED ANNEALING

Optimized Simulated Annealing was evolutionary created by continuing the work of Hu and Marculescu. Their Simulated Annealing and Branch and Bound algorithms are available through the NoCmap project SLD (2010). We ported their two algorithms into our UNIfied framework for optimization and evaluation of NoC application MAPping algorithms (UniMap) Radu (2010). UniMap's goal is to allow the evaluation and potential optimization of different application mapping algorithms for NoCs, under a common frame Radu and Vinţan (2010). OSA is a Simulated Annealing approach that is optimized for the Network-on-Chip application mapping problem, by applying and adapting some of the best practices for Simulated Annealing used in task mapping problems Orsila et al. (2008).

The pseudocode for the Optimized Simulated Annealing is presented in Algorithm 1. It is derived from the general Simulated Annealing proposed by Orsila et al. (2008).

Algorithm 1 Optimized Simulated Annealing (OSA)

```
Require: $M_i \neq 0$
Ensure: $T_i \geq 1$
  $M \leftarrow M_i$
  $C \leftarrow BitEnergyCost(M_i)$
  $M_{best} = M$
  $C_{best} = C$
  $T_f = 0.001$
  $R = 0$
  for $i = 0$ to $\infty$ do
    if $i \% L = 0$ then
      $R = 0$
    end if
    $T = T_i \cdot 0.9^{\lfloor \frac{i}{L} \rfloor}$
    $M_{new} = PDFbasedSwapping(M, T)$
    $C_{new} = BitEnergyCost(M_{new})$
    $\Delta C = C_{new} - C$
    if $\Delta C < 0$ or $NormInvExpAccept(\Delta C, T)$ then
      if $C_{new} < C_{best}$ then
        $M_{best} = M_{new}$
        $C_{best} = C_{new}$
        $R = 0$
      else
        $R = R + 1$
      end if
      $M = M_{new}$
      $C = C_{new}$
    else
      $R = R + 1$
    end if
    if $T \leq T_f$ and $R = L$ then
      break
    end if
  end for
  return $M_{best}$
```

We start by describing how OSA implements the key issues of a typical SA algorithm. Then, we conclude by summarizing our Optimized Simulated Annealing.

3.1 *Mapping cost*

The mappings produced with OSA are evaluated in terms of energy consumption, using the simple analytical model from Hu and Marculescu (2003).

3.2 *Annealing schedule*

We have chosen for OSA the geometric annealing schedule because this is the most used and recommended one Fouskakis and Draper (2007) and because the general SA implementations use it too. The geometric annealing temperature schedule defines the temperature at iteration i as: $T = T_0 q^{\lfloor \frac{i}{L} \rfloor}$. T_0 is the initial temperature and L is the number of iterations per temperature level. We set q, the geometric progression ratio, to 0.9, because this value is also used in Kirkpatrick et al. (1983) and SLD (2010).

OSA uses an initial temperature set to 1 by default but, this is considered a parameter of the algorithm. The final temperature is fixed to 0.001. Hu and Marculescu's SA starts from a temperature of 100 and the final temperature is not bounded (a different stop criteria is implemented).

3.3 *Number of iterations per temperature level*

The general Simulated Annealing sets $L = 100(N \times N)^2$, where $N \times N$ represents the size of the 2D mesh. For example, for a 4x4 2D mesh, SA tries $L = 25600$ mappings, at each temperature level. These mappings are randomly generated, from the current mapping, by interchanging two cores, or by placing a core into an empty NoC node. If we consider SA runs for 100 temperature levels, this leads to 2560000 mappings generated. An Intel Xeon processor from our HPC system ULBS (2010) evaluates a mapping in 0.04 ms. Thus, for this example, SA would run for about 102 seconds. On a 10x10 mesh, SA would require more than one hour running time. However, the general SA algorithm is not bounded by the number of temperature levels, and we observed that it easily runs for more than 150 levels of temperature for a 4x4 2D mesh. We argue that this number of iterations per temperature level is very high and it has a deep impact on SA's speed. Also, we observe that this number is only NoC aware. It is by no means application aware. For example, mapping 15 or 16 cores on a 4x4 2D mesh uses the same L.

Considering the above simple observations, we use in OSA the following relation to compute the number of iterations for each temperature level:

$$L_{OSA} = {}_nC_2 - {}_{n-c}C_2 = \frac{c(2n-c-1)}{2}, c, n \in \mathbb{N}, n \geq c \quad (1)$$

n is the number of NoC nodes and c is the number of cores to be mapped.

This number of iterations per temperature level represents how many distinct core swappings are possible for a given NoC with n nodes and c cores, so that at most two cores change positions in comparison with the initial mapping.

It may easily be observed that:

$$\max\{L_{OSA}\} = \frac{n(n-1)}{2} \overset{not.}{=} L_{OSA_{max}} \quad (2)$$

This happens when the number of cores to be mapped is equal with the number of NoC nodes.

Recall that SA has $L = 100(N \times N)^2$. Because we noted the number of NoC nodes with n, we can write that $L_{SA} = 100n^2$. It is obvious that $L_{OSA} < L_{OSA_{max}} < L_{SA}$. Also,

$$\left.\begin{matrix} O(L_{OSA})=O(n^2) \\ O(L_{SA})=100O(n^2) \end{matrix}\right| \Rightarrow 1 - \frac{O(L_{OSA})}{O(L_{SA})} = 99\% \quad (3)$$

This is in perfect concordance with our further quantitative results.

L_{OSA} counts all mappings that are obtained by making a single modification (core swapping) to the given mapping (otherwise, the total possible mappings are of course ${}_nP_k > L_{OSA}$). All the mappings that derive from a given mapping in this way, are what we call the mapping's immediate neighborhood. We can make an analogy with Markov chains. OSA's number of iterations per temperature level can be associated with the number of possible single step transitions from a Markov chain, which describes the mapping state space exploration performed by OSA. Later on, we will show how OSA assigns probabilities for each single step transition, using the concept of Probability Distribution Function (PDF). One may also observe that OSA's number of iterations per temperature level is both NoC and application aware: n is NoC topology characteristic and c is application characteristic. Returning to the example with the 4x4 2D mesh, we compute L_{OSA}, for mapping 16 cores, to be 120. This means a run time of 0.48 seconds. Compared with how much time the general SA needs (102 seconds), we get a speedup of 99.53%. For the 10x10 2D mesh, $L_{OSA} = 4950$. This means a runtime of 19.8 seconds and a speedup of 99.5%.

Some criticism of this approach might be that, although huge speedup can be obtained with L_{OSA}, OSA might find worse solutions because it does less exploration of the search space. We argue that the general SA can easily repeat a lot of moves without finding better solutions. Additionally, OSA considers the initial temperature a parameter, which can be increased so that the algorithm does more exploration. We will sustain our assessments through simulations.

3.4 Acceptance function

Both general SA and CSA algorithms use the Metropolis acceptance probability function: $P(\Delta C) = e^{-\frac{\Delta C}{KT}}$. OSA however uses the normalized inverse exponential acceptance function because it is shown in Orsila et al. (2008) that it yields better results when it is used for task scheduling. This function is defined as:

$$P(\Delta C) = \frac{1}{1 + e^{-\frac{\Delta C}{C_0 T}}} \quad (4)$$

The practical difference between the two functions is that the exponential form always accepts a new mapping that is equally good compared to the current mapping, whereas the (normalized) inverse exponential form accepts such a new mapping with a 50% probability.

OSA's acceptance function performs cost normalization by dividing the cost variance (ΔC) with the initial mapping cost (C_0). This allows the temperature T to be independent of the cost function: $T \in (0, 1]$. Note that OSA sets the initial temperature to 1 and the final temperature to 0.001 but, it allows that $T_i > 1$. Regarding the normalized inverse exponential function, this would mean the initial cost is artificially increased.

Knowing OSA's final temperature and acceptance function, we can calculate that OSA considers the system frozen when the energy consumption varies with less than 0.5%.

3.5 PDF-based swapping

The SA algorithms mentioned in our related work use random core swapping: a core is selected randomly and this core is then swapped with another randomly selected core (an empty node can also be used). OSA does not use a uniformly random probability when determining the core to be moved. Instead, it adapts the variable grain single move (based on probability densities and used for task mapping Orsila et al. (2008)) into a variable grain swapping move, which uses two Probability Density Functions. OSA builds a PDF for each core, based on the amount of data communicated by it. This leads to better chances for selecting a core that communicates more data, than a core which communicates less data. As the annealing temperature decreases, the probabilities uniformly equalize. Therefore, at low temperatures, all cores have an equal chance to get selected for swapping. Through this approach, OSA uses problem knowledge (dynamic characteristics) to explore the search space.

$$P[SelectedCore = i] = \frac{1}{c} + \frac{T}{T_i}\left(\frac{coreToComm_i}{totalToComm} - \frac{1}{c}\right) \quad (5)$$

- c is the number of cores to be mapped;
- T and T_i are the current and initial temperatures;

- $coreToComm_i$ is the amount of data communicated by core i;
- $totalToComm$ is the total data communicated by all cores.

The second core used for swapping is selected by accounting for the communication volumes between the core to be swapped and the rest of the cores. Each core gets such a PDF associated before the annealing starts.

$$P[c_i \leftrightarrow c_j] = \frac{comm_{ij}}{totalComm} \tag{6}$$

- $comm_{ij}$ is the communication volume between cores i and j (this value is positive if core i sends data to core j, or core j sends data to core i; otherwise, it is zero);
- $totalComm$ is the data amount communicated by the entire application.

According to the PDF described above, the second core is selected. Then, OSA searches, in a uniformly random way, for a direct neighbor of the second selected core. This one will be swapped with the first core. This kind of move tries to make communicating cores to attract each other, to cluster themselves, in a natural manner.

Compared to CSA, our algorithm clusters the cores dynamically, during the annealing phase. OSA does not work with predetermined clusters, and it also does not cluster the NoC nodes. Network-on-Chip node clustering is not needed because OSA looks in the NoC node's neighborhood.

We call this kind of move a *PDF-based swapping* move. At every temperature level, OSA performs exactly L PDF-based swappings.

3.6 Stopping condition

OSA uses the stopping function recommended by Orsila et al. (2008) and called coupled temperature and rejection threshold. The annealing process stops when the temperature reaches or goes below the final temperature and the last L tried mappings did not lead to a solution better than the best solution found so far. Therefore, OSA runs for a determined number of annealing temperatures, which can be exceeded while better solutions are found.

Because OSA's stopping condition determines a number of annealing levels independent of the problem size, the runtime of our algorithm is quasi-constant when the algorithm is run more than once, in exactly the same conditions. This property does not apply to Hu and Marculescu's Simulated Annealing.

3.7 Summary

OSA starts from an initial mapping, M_i, randomly generated. Another input parameter can be the initial temperature, T_i, which is set by default to 1. A bit energy model is used by OSA to evaluate the mappings. We use a standard geometric annealing schedule, with L_{OSA} annealing iterations per temperature level. The move function is an adequate swapping based on Probability Density Functions. A normalized inverse exponential function determines when worse mappings are accepted. OSA stops when the final temperature ($T_f = 0.001$) is reached and the number of consecutive rejected moves, R, reaches L. While in Orsila et al. (2008) R counts how many moves were rejected since the last accepted move, in OSA we use R to count how many moves were rejected, per temperature level, since the last current best mapping found. This means that irrespective of the final temperature, OSA keeps running while it still finds better mappings. The Simulated Annealing from Orsila et al. (2008) needs to wait until the number of unaccepted moves, counted from the last one accepted, reaches L. OSA's stopping condition is therefore more coupled to T_f. This makes OSA's number of iterations to be independent of the NoC topology and tiles number. Since we consider the energy variations are small enough when the final temperature is reached, we believe our way of computing R is more suitable for a Simulated Annealing applied to NoC application mapping.

Compared to the general SA, our algorithm determines the number of iterations per temperature level by computing the neighborhood size of the current mapping. Also, OSA moves from one mapping to another using an implicit clustering technique, based on Probability Density Functions. This kind of clustering is implicit and dynamic, whereas CSA's clustering approach is explicit and static.

Currently, OSA works only with 2D mesh topologies but, it can be adapted to work with other NoC topologies, too. Like Hu and Marculescu's SA, OSA is also capable to generate the routing functions, in a deadlock- and livelock-free manner, and to check if the obtained mapping meets the bandwidth constraints.

4. SIMULATION METHODOLOGY

Network-on-Chip benchmarking is still an open problem. The Open Core Protocol International Partnership (OCP-IP) currently works with some of the most prestigious NoC research groups from the world to build a suitable benchmarking methodology for Network-on-Chip simulation Salminen et al. (2010).

We use in this research the E3S benchmark suite Dick (1998). It includes Communication Task Graphs (CTGs) Marculescu et al. (2009) for five classes of real applications: automotive, consumer, networking, office and telecommunications. Because each application has only a few cores, we combined all Application Characterization Graphs (APCGs) Marculescu et al. (2009) from each application domain and considered them to form a single system of several similar applications that need to be mapped on the same Network-on-Chip.

Additionally, we work with some of the most used APCGs found in literature: Multimedia System - MMS (CTG 0) Hu and Marculescu (2005), MMS (CTG 1) Hu and Marculescu (2003), Video Object Plane Decoder - VOPD (CTG 0) Murali and Micheli (2004a) and several core graphs from SUNMAP Murali and Micheli (2004b): Picture in Picture (PIP), MPEG4, Multi-Window Display (MWD) and VOPD (CTG 1). We also introduce the H.264 decoder, with the tasks obtained through data partitioning - H.264 (CTG 0) and through functional partitioning - H.264 (CTG 1) van der Tol (2003), by manually creating the APCG from the CTG.

We consider the most common Network-on-Chip architecture: a 2D mesh with regular tiles, using wormhole switching and XY routing. The NoC topology size is a simulation parameter. The NoC link bandwidth was set sufficiently high so that bandwidth constraints are always met. The energy required to transfer a bit of data was taken from NoCmap.

Table 1 summarizes the benchmarks used in this research. It also presents the NoC 2D mesh size used for mapping each benchmark.

Table 1. Benchmarks

Benchmark	# cores	# NoC nodes	NoC size
auto-indust	24	25	5 × 5
consumer	12	12	4 × 3
networking	13	16	4 × 4
office-automation	5	6	3 × 2
telecom	30	30	6 × 5
PIP	8	9	3 × 3
MPEG4	12	12	4 × 3
MWD	12	12	4 × 3
H.264 (CTG-0)	14	16	4 × 4
H.264 (CTG-1)	16	16	4 × 4
VOPD (CTG-0)	16	16	4 × 4
VOPD (CTG-1)	12	12	4 × 3
MMS (CTG-0)	16	16	4 × 4
MMS (CTG-1)	25	25	5 × 5

In order to increase the simulations' accuracy we mapped each application 1000 times, with each algorithm (SA, OSA and BB). For each simulation, the initial mapping was randomly chosen. To make the comparisons fair, we set the seed of the random number generator, so that all algorithms start from the same search space point, every simulation.

For each mapping, we recorded the mapping, its cost (in pJoule), the runtime of the algorithm (user CPU time) and the average heap memory consumption.

5. RESULTS

In this section, we evaluate our Optimized Simulated Annealing by comparing it with the two algorithms from NoCmap: Simulated Annealing and Branch and Bound.

5.1 Runtime

Figures 1 and 2 present a runtime comparison between OSA and SA and respectively, OSA and BB. The speedups are obtained as an average of the 1000 runtime speedups, per benchmark.

Figure 1 shows that OSA is much faster than Hu and Marculescu's Simulated Annealing. We obtained an average speedup of 98.95%. This result is corelated with our expectations derived from the comparison between L_{SA} and L_{OSA}. The "lowest" speedups are on office-automation and PIP, the benchmarks with the smallest number of IP cores. We justify this significant speed gain mainly by the way OSA computes the number of iterations per temperature level.

It can be seen in Figure 2 that OSA is slower than BB by 24%, on average. However, for half of the benchmarks, OSA is faster. Compared to Branch and Bound, our algorithm obtained poor runtimes on MPEG4 (more than twice slower), H.264 (1.5 times slower in both cases) and lower but comparable runtimes for PIP, office-automation, VOPD (CTG 1) and auto-indust. We also observe OSA was faster on the biggest benchmarks: 25% speedup for MMS (25 cores) and 41% speedup for telecom (30 cores).

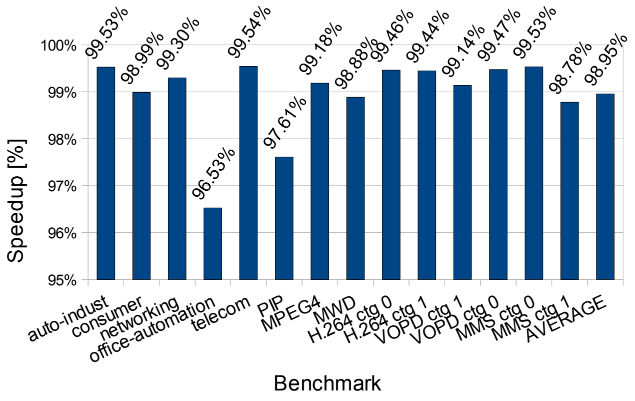

Fig. 1. OSA speedup over SA

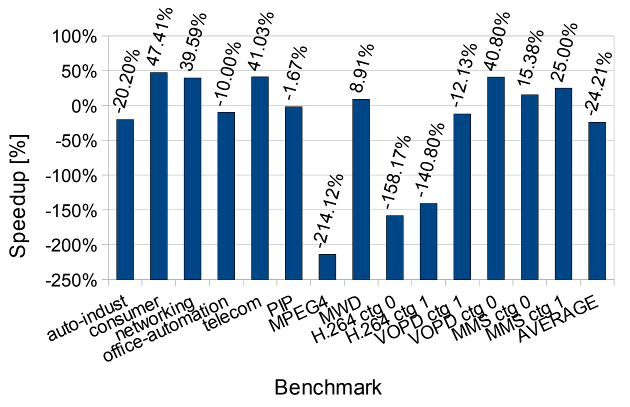

Fig. 2. OSA speedup over BB

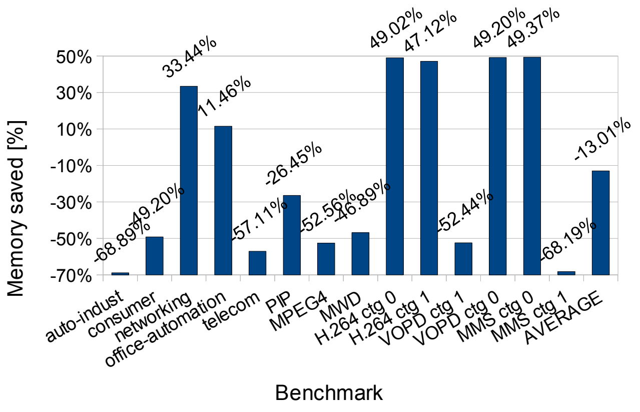

Fig. 3. OSA compared to SA in terms of heap memory consumption (a positive value means OSA consumes less memory)

5.2 Memory

Next, we see how OSA's memory consumption is, compared to the memory consumed by Simulated Annealing and Branch and Bound.

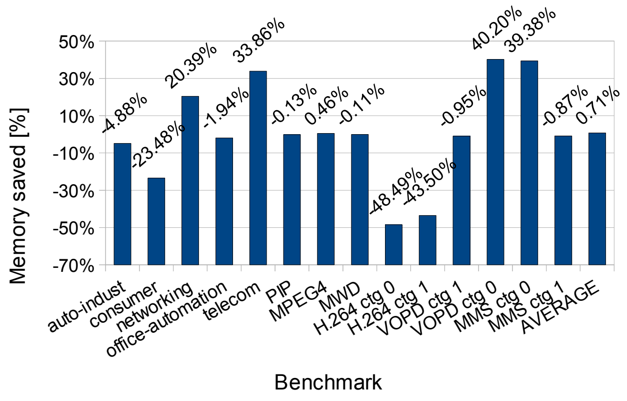

Fig. 4. OSA compared to BB in terms of heap memory consumption (a positive value means OSA consumes less memory)

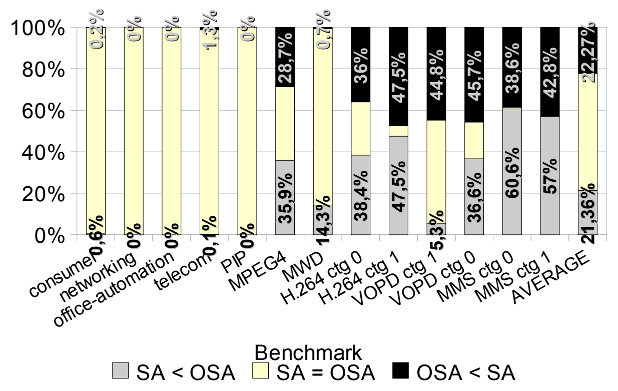

Fig. 5. OSA mappings cost, compared to SA mappings cost

Simulated Annealing consumes less memory than OSA (see Figure 3), when mapping the benchmarks with more than 16 cores. OSA manages to beat SA on several benchmarks with 16 cores but, on average, Simulated Annealing consumes with 13% less memory than our Optimized Simulated Annealing.

However, compared to Branch and Bound, OSA takes a little bit less memory on average. Actually, Figure 4 points out the tendency of Branch and Bound to grow its memory requirements as the problem size gets higher: OSA consumes with more than 33% less heap memory than BB, on telecom.

5.3 Solution quality

This section presents the quality of the solutions found by the three algorithms.

Figure 5 compares the mappings found by SA and OSA. For each benchmark, we evaluate the 1000 mappings returned by the two algorithms and count how many times one algorithm returned better mappings than the other one (marked with "<" in the chart's legend). Cases when both algorithms returned mappings with exactly the same cost are marked distinctively. We notice that both algorithms find the same "best solution", after all 1000 runs, for benchmarks: networking, office-automation and PIP. For the last two of these three benchmarks, we confirm

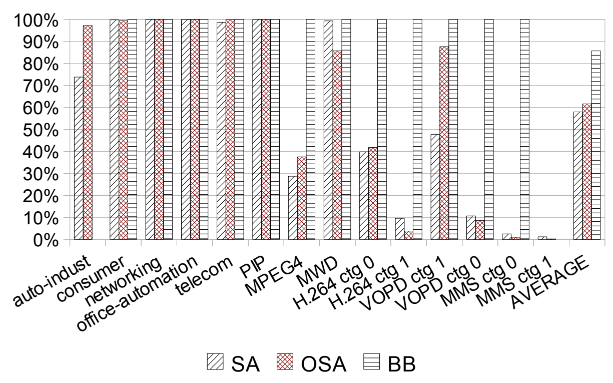

Fig. 6. Best solution percentage

the solution is optimal because we applied an Exhaustive Search. Overall, OSA finds worse solutions than SA for 6 of the 14 benchmarks used in our simulations.

We also compared only the best solutions found by each algorithm. We find out that SA and OSA always find the same best solution. However, Branch and Bound fails to obtain the best mapping found by SA and OSA in two cases: for MMS (CTG 1), the energy lost with BB's mapping is 0.1% and for auto-indust the energy loss is around 6%.

Figure 6 shows how many times the best solution, given by all three algorithms, was found by each one of them. OSA finds the best solution more often than SA for several benchmarks: auto-indust, telecom, MPEG4, H.264 (CTG 0), VOPD (CTG 1). BB outmatches OSA for the MMS benchmarks, VOPD (CTG 0), H.264 (CTG 1), MWD and consumer. Another observation is related to BB: it finds the best solution with probability 1 for all benchmarks, except auto-indust and MMS (CTG 1). We also observe that, on average, OSA finds the best solution a bit more frequently than SA.

We also averaged the quality of the 1000 mappings per benchmark. Branch and Bound is the algorithm that, on average, gives the mapping with the smallest energy consumption. It fails just on auto-indust benchmark, where OSA provides the best average mapping cost. Optimized Simulated Annealing achieves for MMS (CTG 1) a far better average cost compared to Simulated Annealing: more than 34% energy gain is obtained. For the rest of the benchmarks, the differences between OSA and SA are less than one percent. Compared to BB, OSA provides solutions that are worse with no more than 2.5% on each benchmark, except auto-indust, where OSA is better with more than 6% than Branch and Bound.

Using 1000 simulations per benchmark, we have previously shown that the percentage of better solutions was lower for OSA than for SA on six benchmarks: MPEG-4, MWD, H.264 (CTG-0), MMS (both CTGs) and consumer. In order to get OSA's percentage of better solutions over SA's percentage, we reran OSA over the mentioned benchmarks, with an increased initial temperature. Raising the initial temperature allows OSA to evaluate more mappings. Also, the higher the temperature is, the bigger is the probability to accept "bad" moves during the annealing process. Table 2 shows what initial temperature was

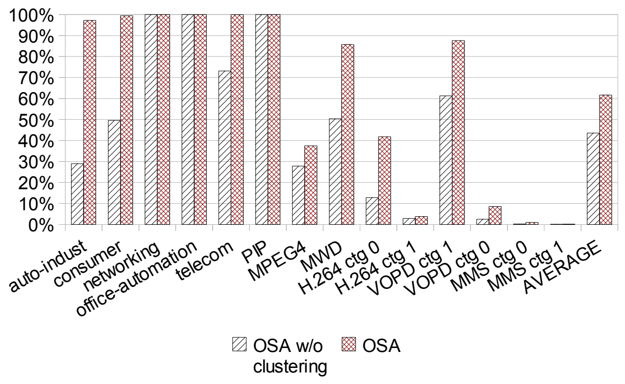

Fig. 7. The influence of OSA's clustering on best solution percentage

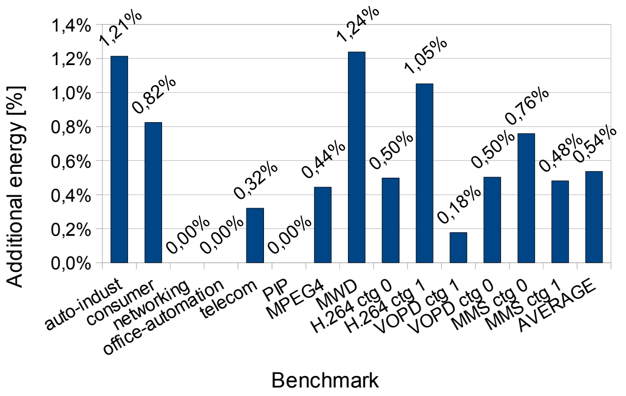

Fig. 8. Additional energy consumed by the mappings obtained with OSA without clustering

needed to achieve our goal and what was the speedup obtained with this increased initial temperature. Note that we raised exponentially OSA's initial temperature due to OSA's geometric annealing schedule. By increasing the initial temperature, we managed to make OSA better than SA on all six benchmarks, except one. For MMS (CTG 1), we were only able to reduce the difference between SA and OSA to half. Still, for MMS (CTG 1), OSA gives significantly better solutions on average.

Table 2. OSA initial temperature and speedup over SA

Benchmark	Speedup (%)	Initial temperature
MPEG4	97.51	1e10
MWD	96.76	1e10
H.264 (CTG-0)	99.18	1e2
MMS (CTG-0)	97.41	1e17
consumer	98.91	2

5.4 The importance of clustering

In order to illustrate how important OSA's clustering technique is, we present next a comparison between OSA with and without clustering. The single thing that distinguishes OSA without clustering from OSA (with clustering) is that, in the first case, the simple random core swapping is used, without any restrictions.

Figure 7 shows how frequently the best solution is found. For all benchmarks, OSA with clustering finds the best solution more frequently than OSA without clustering. More that this, we observe significant differences for the benchmarks mapped onto the 4x4, 5x5 and 6x5 2D mesh NoCs. It is important to mention that the two OSA variants find the same best solution for all benchmarks, except MMS (CTG 1), where the best mapping found by OSA w/o clustering is 0.02% worse. On average, the best solution percentage for OSA with clustering is 18% higher than for OSA without clustering.

Figure 8 shows how much energy is consumed, on average, by OSA without clustering (compared to OSA with clustering). It may be noticed that, for all benchmarks, OSA without clustering generates mappings that determine additional energy consumption. The clustering technique leads to a lower energy consumption with more than 1% in some cases. OSA with clustering always gives better average results (more than half a percent) than OSA without clustering.

5.5 Simulations on bigger 2D meshes

We used four instances of the VOPD benchmark with 16 cores and we mapped them on an 8x8 2D mesh. SA run ten times and OSA and BB run 100 times. We obtained an average running time of 12.65 hours (per simulation) for SA. BB required just 114 seconds and OSA was 36% slower than BB. Still, OSA's runtime is significantly lower than CSA's runtime: 4750 seconds Lu et al. (2008). OSA consumes with approximately 39% less memory than Branch and Bound. The best mapping was found by Simulated Annealing. However, OSA's best mapping is only 0.7% worse. We obtained a BB average mapping cost 70% worse than OSA's.

Using all E3S benchmarks we obtained 84 cores that we mapped onto a 10x9 mesh. SA required approximately 70 hours per simulation. Branch and Bound needed only 380 seconds (on average). OSA was 48% slower than BB. OSA consumed approximately the same memory Branch and Bound required (we noticed Branch and Bound prunes 85% to 93% of the explored search space). Branch and Bound gave, on average, a mapping cost 76% worse than OSA's. Simulated Annealing found the best solution but, it is better than OSA's best solution by only 0.09%.

Using the H.264 (CTG 1), MMS (CTG 0), MMS (CTG 1), MPEG4, MWD and VOPD (CTG 0) benchmarks, we obtained 97 cores (10x10 NoC). We used only OSA and BB (both were run ten times). Optimized Simulated Annealing run on average approximately 15.9 minutes per simulation, being only 3% slower than BB. The memory consumption was similar (40 MB). The solution quality was the same, like in the previous case.

By combining all non E3S benchmarks, we get a benchmark with 131 cores (12x11 NoC). OSA required, on average, approximately 51 minutes mapping this application. Branch and Bound was 15% faster. In this case, OSA consumed less memory, 36 MB, while BB memory requirements were 14% higher. Optimized Simulated Annealing found each time a better mapping. OSA's solutions are 79.4% better than BB's solutions.

Finally, we combined all of our benchmarks and obtained 215 cores (15x15 NoC). OSA run for 8.4 hours and consumed (on average) 265 MB of memory, for each simulation. BB's runtime was more than half OSA's runtime. Memory consumption was also significantly lower: only 158 MB. However, all mapping attempts with Branch and Bound failed. Each time, BB did not finish mapping because it pruned more than 98.7% of the search space. We notice BB's memory consumption does not grow exponentially but, the quality of solution is heavily affected, up to the point where no solution is found.

6. CONCLUSIONS AND FURTHER WORK

We presented in this paper an Optimized Simulated Annealing (OSA) for Network-on-Chip application mapping. Like Hu and Marculescu's Simulated Annealing, OSA is energy- and performance-aware. In contrast with their work, our approach uses application knowledge. Like Clustered-based Simulated Annealing, OSA also performs clustering but, implicitly and dynamically, not explicitly and statically, with certain performance benefits. OSA proves to be much faster than SA. In accordance with our theoretical expectations, we obtained an average runtime speedup of 98.95% while the quality of the mapping solution was not lost. We showed OSA is feasible for NoC meshes with size higher than 10x10. OSA is also comparable to Branch and Bound in terms of memory consumption and speed. However, Branch and Bound failed to find better solutions on auto-indust and MMS (CTG 1) benchmarks. More than this, the mapping solution given by BB is more than 70% worse than the one found by OSA, when mapping cores onto a 2D mesh with size greater than 8x8. We also found that Branch and Bound is unable to map an application with 215 cores, onto a 15x15 NoC.

The comparisons between OSA and CSA runtimes are likely to be unfair. This can be due to several reasons: implementation language (OSA is written in Java but, we do not know how CSA is implemented), cost function (OSA is energy aware, while CSA is bandwidth and latency constrained). Also, CSA does not specify the number of generations per temperature level. Still, we consider OSA is faster than CSA because of the significantly high differences in runtime.

As further work, we plan to implement CSA in UniMap. Other directions for future work are to evaluate OSA using bandwidth constraints and generating deadlock- and livelock-free routing functions. We also intend to evaluate OSA on topologies other than the 2D mesh.

ACKNOWLEDGEMENTS

This research was supported by CNCSIS no. 485/2008 research grant offered by the Romanian National Council for Academic Research and by POSDRU financing contract POSDRU 7706.

REFERENCES

Dick, R. (1998). The embedded system synthesis benchmarks suite (E3S) website. URL `http://ziyang.eecs.umich.edu/~dickrp/e3s/`.

Fouskakis, D. and Draper, D. (2007). Stochastic optimization: a review. *International Statistical Review*, 70(3), 315–349. doi:10.1111/j.1751-5823.2002.tb00174.x.

Hu, J. and Marculescu, R. (2003). Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, 233–239. ACM, Kitakyushu, Japan. doi:10.1145/1119772.1119818.

Hu, J. and Marculescu, R. (2005). Energy- and performance-aware mapping for regular NoC architectures. *IEEE Transactions on ComputerAided Design of Integrated Circuits and Systems*, 24(4), 551—562.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671–680.

Lu, Z., Xia, L., and Jantsch, A. (2008). Cluster-based simulated annealing for mapping cores onto 2D mesh networks on chip. In *Proceedings of the 2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, 16. IEEE Computer Society, Washington, DC, USA. doi:10.1109/DDECS.2008.4538763. ACM ID: 1545632.

Marculescu, R., Ogras, U.Y., Peh, L., Jerger, N.E., and Hoskote, Y. (2009). Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, 28(1), 3–21.

Murali, S. and Micheli, G.D. (2004a). Bandwidth-Constrained mapping of cores onto NoC architectures. In *Proceedings of the conference on Design, automation and test in Europe - Volume 2*, 20896. IEEE Computer Society.

Murali, S. and Micheli, G.D. (2004b). SUNMAP: a tool for automatic topology selection and generation for NoCs. In *Proceedings of the 41st annual Design Automation Conference*, 914–919. ACM, San Diego, CA, USA. doi: 10.1145/996566.996809.

Orsila, H., Salminen, E., and Hamalainen, T.D. (2008). Best practices for simulated annealing in multiprocessor task distribution problems. In *Simulated Annealing*, 321–342. I-Tech Education and Publishing KG.

Radu, C. (2010). Unified framework for Network-on-Chip application mapping (PhD application). URL `https://code.google.com/p/unimap/`.

Radu, C. and Vinţan, L. (2010). Optimizing application mapping algorithms for NoCs through a unified framework. In *Roedunet International Conference (RoEduNet), 2010 9th*, 259 – 264. IEEE Computer Society, Sibiu, Romania.

Salminen, E., Srinivasan, K., and Lu, Z. (2010). OCP-IP network-on-chip benchmarking workgroup. URL `http://www.ocpip.org/uploads/dynamic_areas/Cv8XdaKTKDztFpWKPqsl/6189/NoC%20Working%20Group%20Overview%20WP.pdf`.

SLD (2010). NoCmap: an energy- and performance-aware mapping tool for Networks-on-Chip. URL `http://www.ece.cmu.edu/~sld/wiki/doku.php?id=shared:nocmap`.

ULBS (2010). ULBS HPC cluster. URL `http://zamolxe.hpc.ulbsibiu.ro/`.

van der Tol, E.B. (2003). Mapping of h.264 decoding on a multiprocessor architecture. In *Proceedings of SPIE*, 707–718. Santa Clara, CA, USA. doi:10.1117/12.476234.