Title
Energy Aware Mapping for Reconfigurable Wireless MPSoCs

Permalink
https://escholarship.org/uc/item/17b0d3v2

Authors
Hussien, AMA
Amin, R
Eltawil, AM
et al.

Publication Date
2014-03-24

DOI
10.1109/TVLSI.2014.2309013

Peer reviewed
Energy Aware Mapping for Reconfigurable Wireless MPSoCs

Amr M. A. Hussien, Rahul Amin, Ahmed M. Eltawil, and Jim Martin

Abstract—Energy management for multimode software defined radio systems remains a daunting challenge. This brief develops a high level framework that generates a multiprocessor systems on chip architecture from a library of heterogeneous processing resources that can be reconfigured to support various modes of operation. The framework proposes joint task and core mapping with system level floorplanning. With the objective of minimizing energy, we develop an analytical probabilistic model that considers static, dynamic, configuration, and communication energy components for multiple applications characterized by probabilities of execution. Finally, a fast energy aware joint task and core mapping heuristic is proposed and performance is demonstrated on realistic benchmarks.

Index Terms—Core mapping, multiprocessor systems on chip (MPSoC), synthesis, task mapping.

I. INTRODUCTION

The rich space of applications/configurations required of modern wireless systems creates a wide range of scenarios that mandates an energy efficient reconfigurable platform. Recently, multiprocessor systems on chip (MPSoC) architectures have evolved rapidly in the race to flexible high-performance embedded computing. This is particularly true for multimode communication systems that require high flexibility and low power simultaneously. Toward that end, numerous techniques have been developed to optimize energy consumption. A key realization that energy aware techniques utilize is that efficiency is highly related to the nature of the application. Usually, the set of target applications can be characterized stochastically where each type of application is represented by a certain execution probability. These probabilities can be obtained through statistical information collected from each user regularly. The consideration of these execution probabilities affects system performance and energy efficiency. For example, a mode with high execution probability should be mapped to lower power processors satisfying the execution requirements to obtain an energy efficient system.

Energy optimization techniques presented in prior work consider only processing energy, or communication energy or both. However, the consideration of reconfiguration energy was neglected. It is important to consider the reconfiguration energy required to switch the processing units (PUs) between different tasks, which would be the expected mode of operation for a multireconfigurable platform that supports different radio access technologies. The reconfiguration cost is highly dependent on the architecture and structure of the platform. For example, if a reconfigurable fabric is used then there is a reconfiguration energy cost associated with the change of configuration bits of configurable logic blocks (CLBs), and connections represented in the switching matrix [1]. In case of multiprocessor-based systems, there is an associated cost in switching context and reloading programs from internal or external memories and so on.

Numerous algorithms have been proposed for task mapping and scheduling [2]–[5], as well as processor mapping to networks-on-chip (NoC) tiles in energy efficient multiprocessor-based systems [6]–[8]. Custom NoC synthesis is presented in [9] to minimize power consumption while satisfying performance constraints. In [10] and [11], a comprehensive survey of task mapping and application mapping onto NoCs is presented.

The work in [12] and [13] considered joint task and core mapping for irregular and custom NoCs assuming a given NoC architecture. Demirbas et al. [14] considered NoC topology generation with task and core mapping to minimize the application latency. In this brief, we consider joint task and core mapping with system level floorplanning to minimize the energy consumption. As an extension to our previous work in [15], we develop a framework that generates a reconfigurable MPSoC architecture from a library of heterogeneous PUs based on a probabilistic model that considers static, dynamic, reconfiguration, and communication energy components for multiple applications characterized by certain execution probabilities.

II. SYSTEM MODEL

The system is assumed to run a set of probabilistic applications, on a heterogeneous platform that comprises different types of PUs connected in a 2-D NoC-based MPSoC. The utilized model is similar to that originally proposed in [4]. In addition to leakage and dynamic processing energy, our proposed model incorporates both communication and reconfiguration energy. Simulation results confirm that exploring the expanded design space leads to design points that are more efficient than the reference baseline technique. To maintain consistency, we adopt the same parameter notations as [4]. In general, the model assumes a set of scenarios $\mathcal{S}$, and each scenario $S_m \in \mathcal{S}$ has an execution probability $p_m$, and includes a set of tasks represented by a directed task graph. Each scenario $S_m \in \mathcal{S}$ is executed for an average time $t_m$. Although the information about scenario execution times may not be present, it can be collected statistically from different users. The set of tasks among all scenarios is defined by $T$, and the set $T_m$ is the set of tasks executed in scenario $S_m$.

The system comprises a set of PU types $\mathcal{P}$, where the $j$th PU type, $P_j$, can have multiple instances such that $\hat{P}_{j,k}$ is the $k$th instance of $P_j$. The maximum possible number of instances $F_j$ of the $j$th PU type is determined at design time based on the task requirements. The static power of $P_j$ is $\sigma_j$. There is a reconfiguration cost associated with switching from one scenario $S_a$ to $S_m$. This cost represents the reconfiguration cost of the PUs to run the tasks in $S_m$, and not originally in $S_a$. The dynamic and reconfiguration powers of the $i$th task, $\delta_{i,j}$, on any instance of $P_j$ is $\sigma_j$. Each task $t_i$ has an execution probability of $p_{i,j}$, average number of executions of $k_{max}^i$, and reconfiguration probability of $r_j$. The value of $r_j$ depends on the transition probabilities among different execution scenarios. The feasibility indicator $f_{i,j}$ is used to identify if it is
feasible to map a task \( t_i \) on \( P_j \), and utilization of running \( t_i \) on \( P_j \) is defined by \( u_{i,j} \).

The communication cost between \( t_i \) mapped to \( \hat{p}_{i,j} \) and \( t_j \) mapped to \( \hat{p}_{q,o} \) depends on the communication volume, and the distance between the cores. Since the cores are potentially heterogeneous, it is necessary to consider custom NoC architectures. At early design stage, the communication energy can be abstracted by point-to-point physical links [9]. Therefore, the communication energy between two cores can be represented by the Manhattan distance. For a specific PU instance \( \hat{p}_{i,j} \) with a width \( w_{i,j} \) and height \( h_{i,j} \), the core location is defined by the \( x-y \) coordinates such that \( (X_{\hat{p}_{i,j}}^i, Y_{\hat{p}_{i,j}}^j) \) are the coordinates of the lower left side, and \( (X_{\hat{p}_{i,j}}^\text{max}, Y_{\hat{p}_{i,j}}^\text{max}) \) are the coordinates of the upper right side. Assuming that routers are attached to the upper right corners of each core, the distance between two PU instances \( \hat{p}_{i,j} \) and \( \hat{p}_{q,o} \) are represented as \( \Delta X_{j,k} = X_{\hat{p}_{i,j}}^\text{max} - X_{\hat{p}_{q,o}}^\text{max} \), and \( \Delta Y_{j,k} = Y_{\hat{p}_{i,j}}^\text{max} - Y_{\hat{p}_{q,o}}^\text{max} \). \( E_{\text{hk}} \) is the link energy per bit.

### III. PROBLEM FORMULATION

This brief considers the problem of building an NoC-based MPSoC architecture from a library of different PU types, followed by joint task mapping to different PU instances, and core mapping with custom NoC floorplanning at design time. The main goal is to minimize the energy consumption including static, dynamic, communication, and reconfiguration energy while maintaining performance constraints. A task \( t_i \) is related to the PU \( \hat{p}_{i,j} \) through task utilization \( u_{i,j} \). Since one PU instance can run more than one task simultaneously, the overall utilization should be satisfied.

The variable \( M_{i,j,k} \) defines the mapping between \( t_i \) and \( \hat{p}_{i,j} \) such that \( M_{i,j,k} = 1 \) if \( t_i \) is mapped on \( \hat{p}_{i,j} \), and 0 otherwise. In addition, the variable \( Z_{m,j,k} \) defines the mapping between scenarios and PU instances such that \( Z_{m,j,k} = 1 \) when any \( t_i \in T_m \) is mapped to \( \hat{p}_{i,j} \) and 0 otherwise. The binary variables \( X_{j,k}^i \) and \( Y_{j,k}^i \) determines the relative position of PU instances \( \hat{p}_{i,j} \) and \( \hat{p}_{q,o} \). The variable \( X_{j,k}^i \) is defined such that \( X_{\hat{p}_{i,j}}^i \geq X_{\hat{p}_{q,o}}^i \), and 0 otherwise. This represents that the PU instance \( \hat{p}_{i,j} \) is completely located to the right of \( \hat{p}_{q,o} \). Similarly, \( Y_{j,k}^i \) is defined such that \( \hat{p}_{i,j} \) is completely located to the top of \( \hat{p}_{q,o} \).

The average energy consumption among all scenarios is derived as

\[
E_{\text{avg}} = \sum_{S_m \in S} Z_{m,j,k} M_{i,j,k} \sum_{t_i \in T_m} \sum_{\hat{p}_{i,j} \in \hat{P}} \left( \sum_{j,k \in \hat{P}} X_{\hat{p}_{i,j}}^i \right) \tag{2a}
\]

\[
+ \sum_{t_i \in T_m} \sum_{\hat{p}_{i,j} \in \hat{P}} \left( \sum_{j,k \in \hat{P}} X_{\hat{p}_{i,j}}^i \right) M_{i,j,k} \sum_{t_i \in T_m} \sum_{\hat{p}_{i,j} \in \hat{P}} \left( \sum_{j,k \in \hat{P}} X_{\hat{p}_{i,j}}^i \right) \tag{2b}
\]

\[
+ \sum_{t_i \in T_m} \sum_{\hat{p}_{i,j} \in \hat{P}} \sum_{\hat{p}_{q,o} \in \hat{P}} \tilde{Q}_{t,t_j} E_{\text{hk}} \left[ WX^{+1,q,o}_{i,j,k} + WX^{-1,q,o}_{i,j,k} + Wy^{+1,q,o}_{i,j,k} + Wy^{-1,q,o}_{i,j,k} \right] \tag{1}
\]

where \( \tilde{Q}_{t,t_j} = \sum_{S_m \in S} Z_{m,j,k} M_{i,j,k} \sum_{t_i \in T_m} \sum_{\hat{p}_{i,j} \in \hat{P}} \left( \sum_{j,k \in \hat{P}} X_{\hat{p}_{i,j}}^i \right) \) is the communication volume between \( t_i \) and \( t_j \) in \( S_m \), and \( \omega_{t,t_j} \) is a weighting factor representing the latency constraint in hops [9] between \( t_i \) and \( t_j \) in \( S_m \). The target is to find the optimal task mapping \( M_{i,j,k} \) as well as the optimal core mapping in terms of the \( x-y \) coordinates \( (X_{\hat{p}_{i,j}}^\text{min}, Y_{\hat{p}_{i,j}}^\text{min}) \) to minimize the average energy consumption.

The mixed integer linear programming (MILP) formulation of the optimization problem is depicted in (2). The constraint in (2b) guarantees that each task is mapped to one and only one feasible PU instance. The constraint in (2c) ensures that the utilization condition is satisfied for each instance in each scenario. The constraint (2d) guarantees that there is no overlapping between two different PU instances. In addition, the four linearization inequalities in (2e) are used to substitute the multiplication of the two binary and continuous terms, where \( D \) is a very large constant.

\[
\min E_{\text{avg}} \tag{2a}
\]

\[
\text{s.t.} \sum_{t_i \in T_m} \sum_{\hat{p}_{i,j} \in \hat{P}} M_{i,j,k} = 1 \quad \forall t_i \in T_m, \quad S_m \in S \sum_{t_i \in T_m} u_{i,j} M_{i,j,k} \leq Z_{m,j,k} \quad \forall S_m \in S \tag{2b}
\]

\[
\sum_{t_i \in T_m} \sum_{\hat{p}_{i,j} \in \hat{P}} \tilde{Q}_{t,t_j} E_{\text{hk}} \left[ WX^{+1,q,o}_{i,j,k} + WX^{-1,q,o}_{i,j,k} + Wy^{+1,q,o}_{i,j,k} + Wy^{-1,q,o}_{i,j,k} \right] \geq 1 \quad \forall \hat{p}_{i,j}, \hat{p}_{q,o} \in \hat{P} \tag{1}
\]

\[
WX^{+1,q,o}_{i,j,k} = \Delta X^{q,o}_{j,k} + D(2 - M_{i,j,k} - M_{i,q,o}) \geq 0 \tag{2d}
\]

\[
WX^{-1,q,o}_{i,j,k} = \Delta X^{q,o}_{j,k} + D(2 - M_{i,j,k} - M_{i,q,o}) \geq 0 \tag{2d}
\]

\[
Wy^{+1,q,o}_{i,j,k} = \Delta Y^{q,o}_{j,k} + D(2 - M_{i,j,k} - M_{i,q,o}) \geq 0 \tag{2d}
\]

\[
Wy^{-1,q,o}_{i,j,k} = \Delta Y^{q,o}_{j,k} + D(2 - M_{i,j,k} - M_{i,q,o}) \geq 0 \forall t_i, t_j \in T \tag{2e}
\]

According to [4], the task mapping problem that considers only static and dynamic power for one scenario and one PU type is reduced to the bin packing problem, which is known to be NP-hard. As a consequence, the problem of joint task and core mapping for multiple scenarios and PU types considering static, dynamic, configuration, and communication energy components is NP-hard in a strong sense.

### IV. PROPOSED DESIGN TIME HEURISTIC ALGORITHM

We propose an iterative heuristic solution to minimize the overall energy consumption. The proposed solution has initial and iterative
mapping algorithms. Each one consists of task mapping, followed by core (re)mapping, where it performs the system level floorplanning to the utilized heterogeneous PU instances.

A. Initial Static Energy Aware Task Mapping

The initial algorithm, shown in algorithm 1, performs core selection and preliminary task mapping based on a cost function that considers static, dynamic, and reconfiguration energy costs. There is no corresponding available communication cost before initial mapping. The set of tasks with the minimum cost function on $P_j$ is included in $T_j$ and all the tasks in $T_j$ are arranged based on utilization in descending order. This heuristic starts mapping tasks to the PUs with lower static power first. This increases the chance to pack more tasks into the instance with lower static power consumption, and contributes accordingly to the total energy saving. As shown in step 9, the PU types are sorted in ascending order according to the static power consumption. Subsequently, the new energy cost is computed for each $t_i \in T_j$ on each PU instance $\hat{p}_{j,k}$. The new energy cost on each instance consists of static, dynamic, and reconfiguration energy in steps 15–17.

In step 20, the task $t_i$ is mapped to $\hat{p}_{j,k}$ with the minimum total estimated cost. After mapping $t_i$ to $\hat{p}_{j,k}$, the total utilization of $\hat{p}_{j,k}$ for each scenario $S_m$ that host $t_i$, denoted by $U_{m,j,k}$, is calculated in step 21, and the mapping variable is updated in step 22. Finally, an initial core mapping takes place by running Algorithm 2 as depicted by step 25.

The initial solution determines the number of utilized PU instances for each PU type, and the initial core mapping for each utilized PU instance. Accordingly, a preliminary architecture is generated at design time. This architecture is used in an iterative solution seeking further energy reduction.

B. Iterative Remapping Solution

After the initial task and core mapping, an iterative solution is used to remap tasks to reduce the total energy consumption. It is highly similar to the initial mapping, however, it considers the communication energy based on recent core mapping. At each iteration, this solution seeks a PU instance for each task $t_i$ that maximizes the energy saving. The utilized PU instances are considered and sorted in ascending order based on the maximum utilization among the different scenarios. Afterward, a new energy cost is calculated for each task $t_i$ on each $\hat{p}_{j,k}$ based on static, dynamic, reconfiguration, and communication cost, while satisfying the utilization constraint. The resulting average energy consumption is reduced with the increase in the number of iterations and performance approaches the optimal solution.

C. Core (Re)Mapping Solution

After each mapping iteration, the core mapping heuristic in Algorithm 2 is used to locate each utilized PU instance in a specific NoC position such that the communication energy is minimized. This core mapping is based on the most recent task mapping. The set of utilized instances that host tasks are defined by $P^o$. The set of mapped and unmapped PU instances is defined by $P_p$ and $P_u$, respectively. The core mapping is determined based on the average number of transactions between the different instances. The algorithm tends to map the instances with a larger number of transactions as close as possible. This heuristic creates system level floorplanning to build a custom NoC architecture that is well suited for the target heterogeneous platform. The heuristic considers the X–Y plane as a feasible area represented by $R$, as in step 3. As presented in step 5, the core mapping heuristic chooses the PU instance with the highest average number of two-way transactions with other instances as the first one to be placed. It is indicated as $\hat{p}_{j_{0},k_{0}}$, and it is mapped with the left lower side $(X_{j_{0},k_{0}}, Y_{j_{0},k_{0}})$ at $(0, 0)$. As a consequence, the rectangular area bounded by $(X_{j_{0},k_{0}}, Y_{j_{0},k_{0}})$ and $(X_{j_{0},k_{0}}, Y_{j_{0},k_{0}})$, defined by $r$ in step 7, is occupied and removed from the set of feasible positions $R$ as in step 8. Then, other instances are mapped one by one in descending order according to the average cost with the currently mapped instances. The position of the selected instance is determined such that it minimizes the total communication cost with the mapped instances as indicated in step 11. The algorithm repeats this process until all instances are mapped. Whenever a new instance is mapped, it is inserted in the set of mapped instances $P_p$ and removed from the set of unmapped instances $P_u$ as presented by step 13. This can be considered as a 2-D bin packing problem. However, for the scope of this brief, we impose no restriction on the area or aspect ratio of the chip.

The algorithm complexity is based on the number of tasks ($N_t$), number of PU types ($N_p$), number of resulting PU instances ($N_i$), and number of scenarios ($N_s$). The complexity of the initial mapping $O(N_pN_tN_iN_s)$, while the iterative mapping is $O(N_p^2N_tN_iN_s)$.

V. PERFORMANCE RESULTS

This section demonstrates the performance of the proposed heuristic for a set of realistic benchmarks that are widely used in current smart phones; namely WCDMA, denoted by A1, LTE, denoted by A2, JPEG encoder, denoted by A3 and MP3 decoding, denoted by A4. These applications are profiled and divided into individual tasks as presented by the task graphs in Fig. 1. The figure also illustrates the communication volumes.

A total of 11 execution scenarios are assumed, where each one comprises of single or multiple applications with equal latency requirements and different throughputs. The different scenarios with their corresponding execution probabilities are shown in Table I. In addition, we assume a library of six different PU types, where each PU type can have a multiple instances. The chosen PUs are: 1) OpenRISC core; 2) ARM-Cortex-A9; 3) Ultra-SPARC-T1; 4) a large reconfigurable FPGA fabric based on the Xilinx-VirtexII Pro CLBs with a total of 5824 CLBs; 5) a small reconfigurable fabric with 728 CLBs of the same type; and 6) a turbo decoder coprocessor.

Table II presents the specifications of the used PU types.

For the set of given tasks, the task execution times, utilization, and different costs associated with PU types are obtained from [1] and [16]–[22]. The utilization of the different tasks on OpenRISC,
Fig. 1. Task graph of different applications. (a) JPEG encoder task graph. (b) MP3 decoder task graph. (c) WCDMA receiver task graph. (d) LTE receiver task graph. (e) List of the individual tasks.

Table I

description of realistic benchmarks

Table II

description of the used PU types

<table>
<thead>
<tr>
<th>PU type</th>
<th>Vdd (V)</th>
<th>Tech (nm)</th>
<th>Area (mm²)</th>
<th>Freq (MHz)</th>
<th>Static power (mW)</th>
<th>Dynamic power (mW/MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenRISC 1200</td>
<td>1.0</td>
<td>65</td>
<td>0.052</td>
<td>250</td>
<td>0.263</td>
<td>0.025</td>
</tr>
<tr>
<td>ARM-Cortex A9</td>
<td>1.0</td>
<td>65</td>
<td>5.7</td>
<td>830</td>
<td>24.6</td>
<td>0.707</td>
</tr>
<tr>
<td>UltraSPARC T1</td>
<td>1.0</td>
<td>65</td>
<td>7.57</td>
<td>610</td>
<td>51.1</td>
<td>0.633</td>
</tr>
<tr>
<td>Large rec. fabric</td>
<td>1.0</td>
<td>65</td>
<td>40.8</td>
<td>100</td>
<td>312</td>
<td>0.0031/CLB</td>
</tr>
<tr>
<td>Small rec. fabric</td>
<td>1.0</td>
<td>65</td>
<td>5.1</td>
<td>100</td>
<td>39.06</td>
<td>0.0011/CLB</td>
</tr>
<tr>
<td>Turbo dec. core</td>
<td>1.0</td>
<td>65</td>
<td>9.38</td>
<td>100</td>
<td>9.1</td>
<td>0.608</td>
</tr>
</tbody>
</table>

VI. Conclusion

This brief proposes a framework that generates an energy-aware MPSoC platform with corresponding task and core mappings. This brief presents an MILP model for energy-aware joint task and core mapping with system-level floorplanning. The developed framework performs core selection, then it proposes an energy-aware joint task and core mapping heuristic with system-level floorplanning for custom NoC generation. The proposed framework can handle large problem size where it becomes infeasible to use optimal solvers. The framework has been applied to a realistic test case and comparative performance results have been presented.

References


