MPMAP : A High Level Synthesis and Mapping Tool for MPSoCs

Amr M. A. Hussien, Ahmed M. Eltawil
Electrical Engineering and Computer Science
University of California, Irvine
Irvine, California 92697
Email:{ahussien, aeltawil}@uci.edu

Rahul Amin , Jim Martin
School of Computing
Clemson University
Clemson, South Carolina 29634
{ramin@,jim.martin@cs.}clemson.edu

***Abstract*—The design of efficient multiprocessor systems on chip (MPSoC) is a daunting challenge. This challenge is driven by the increasing demand in achieving high performance, low power, and reconfigurable applications onto the same platform. As a consequence, there is an emerging need for fast, and efficient design space exploration, while providing abstract models for a wide range of applications, and types of processing units (PUs).**

Therefore, this paper proposes a framework for a fast MPSoC generation with joint task and core mapping with the objective of minimizing the average power consumption. The proposed framework considers the static, dynamic, reconfiguration and communication power components. A tool for high level MPSoC task and core mapping (MPMAP) is built based on the proposed framework. MPMAP provides a flexible XML interface that provides a high level description of the different PU architectures, and different applications scenarios. Additionally, the paper presents a case study of real-life applications that can be adopted in future heterogeneous wireless systems.

I. Introduction

The future of heterogeneous wireless systems highly demands building low power reconfigurable platforms to support seamless reconfiguration between different application scenarios. Numerous reconfigurable architectures have been proposed spanning different technologies including application specific instruction set processors (ASIPs), field programmable gate arrays (FPGAs), and digital signal processors (DSPs). Recently, multi-processor systems on chip (MPSoC) architectures have evolved rapidly in the race to high performance embedded computing [1]. This is particularly true for multi-mode communications systems that require high flexibility and low power simultaneously. Additionally, MPSoCs have a very promising future with the aid of technology scaling.

However, building an efficient reconfigurable MPSoC is still an open issue, and there is an emerging need to develop frameworks that explore the huge design space rapidly and efficiently, with the target of optimizing several design metrics. A common design metric is the power consumption, especially in wireless and portable applications. Towards that end, numerous techniques have been developed to optimize power consumption at algorithm, system, and circuit levels.

The power consumption in MPSoCs is highly dependent on the utilized architectures, and is highly impacted by the task mapping [2], as well as the core mapping. Therefore, there is an extensive work in task mapping with different optimization goals, as well as core mapping optimization. Different task mapping techniques have been developed to optimize the computing performance [3], lifetime reliability [4], and energy consumption [5]. Different methodologies have been used based on simulated annealing (SA) [6], Tabu search (TS) [7], Genetic Algorithms (GA) [8], and Integer linear Programming (ILP) [9]–[11]. The proposed methodologies include both design time techniques [5], [12], and run time mapping techniques [5], [13]. An extensive survey of different task mapping approaches can be found in [14].

Additionally, several work addressed the communication-aware core mapping. In [15], a branch and bound optimization algorithm was proposed to map a set of intellectual property (IP) to NoC with the target of minimizing the energy consumption under specified performance constraints. A summary of different core mapping algorithms is found in [16].

This paper proposes a high level synthesis tool for MPSoC mapping (MPMAP), where both task and core mapping are considered jointly, targeting a set of probabilistic application. The system model proposed in this paper is motivated by the model presented in [5]. However, in this paper, in addition to incorporating static and dynamic energy as in [5], we consider both communication and reconfiguration energy components using an analytic probabilistic model. We then introduce an energy-aware static task mapping heuristic to different PU cores, as well as core mapping to different tiles based on the proposed model.

The paper is organized as follows: Section II presents an overview of the proposed MPMAP high level synthesis tool. Section III presents the application and architecture models. In section IV, the problem formulation is presented. The proposed heuristic is presented in section V followed by the performance results in section VI. Section VII concludes the paper.

II. MPMAP: An overview

MPMAP is a high level synthesis tool for energy efficient MPSoC generation with joint task and core mapping. Figure 1 presents the tool flow, where it uses an XML based interface. The XML specifies the high level architecture and application parameters. The key components in the MPMAP XML specifications are:

1) **Processing model** : It specifies the number of available PU types in the architecture library, as well as the static power values, which are application independent.

2) **NoC model** : It specifies the different energy values of the NoC components such as the link energy per bit, the energy consumed in the router associated with each PU.
3) **Application (scenario) model** : It contains the specifications of the differnt application scenarios such as the number of scenarios, number of task in each scenarios, task execution times, dynamic as well as reconfiguration power values of each task o0n each PU type, task reconfiguration times different PU types, communication volume between different tasks in each scenario...etc.

The main benefit of the XML interface is the easy integration with profiling tools that are used to generate the high level architectural and application parameters such as SimpleScalar, SimplePower, GEM5, McPAT ...etc. The data is extracted from the XML interface through an XML parser and fed into the framework. Then, the framework runs a heuristic algorithm that determines the maximum number of instances of each PU types, based on the application tasks. Then it performs a joint task and core mapping optimization. As a consequence, it generates:

1) An NoC-based MPSoC architecture from the given library of different PU types. Each PU types can have one or more instances in the resulting architecture.
2) Task mapping of each task to a certain PU instance.
3) Mapping of each PU instance (core) to a certain tile in the NoC.

The inputs to MPMAP are provided by other profiling tools and statistical measurements. While the MPMAP presents a high level characterization of application and architectures, the used model can easily fit to several application sets including, but not limited to, streaming applications, and software defined radios (SDR).

III. System Model

The system model assumes a generic platform consisting of a heterogeneous MPSoC architecture with different types of PUs. For the sake of generality, we impose no restrictions on the types of PUs, for example, a PU can be a general purpose processor (GPP) unit, a DSP unit, an ASIP unit, or a reconfigurable fabric that runs a task in hardware. Each PU can run more than one task simultaneously based on its computational capability and the computational requirements of these tasks. The PUs are communicating through available means of on chip communications. In particular, we assume communication through 2D network on chip (NoC) grid, where each PU instance is located to a certain tile in the NoC, as shown in Fig. 2. Communication between two tasks is performed through a number of data transactions taking place between the corresponding PUs. When a certain application comprising a set of tasks starts to run on the platform, the tasks are mapped to different PUs based on a mapping procedure, and the selected PUs are configured to perform the corresponding tasks. Fig. 2 illustrates the idea of mapping an application, represented by the task graph, to different PU types, as well as mapping the PU cores to different NoC tiles.

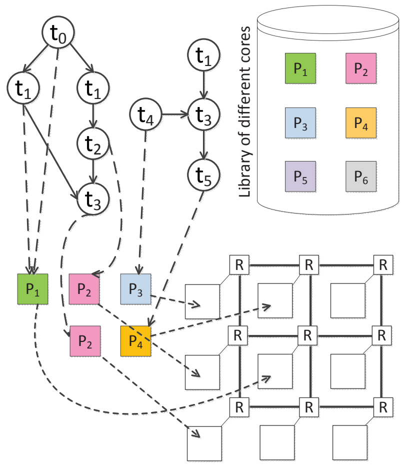


Fig. 2: Task and Core mapping

The system can operate in one of 3 states: {**IDLE**, **ACTIVE**, and **RECONFIG**}. In the IDLE mode, only static power is consumed. In the case of ACTIVE mode, the static, dynamic, and communication powers are considered. The RECONFIG mode takes place upon reconfiguration from one scenario to another. In this case, static, and reconfiguration power are considered. Our solution is based on the framework originally proposed in [5], however our contribution is to increase the dimensionality of the problem by including both communication and reconfiguration energy, in addition to leakage and dynamic energy. We further consider the core mapping to minimize communication energy.

A. Application model

The model assumes that the system supports a different set of applications, where each application can operate at different modes of operations. Different modes for different applications construct different scenarios, and one scenario $S_m \in \mathcal{S}$ is executed at a time, where $\mathcal{S}$ is the set of all possible scenarios and S_m represents the m^{th} scenario. Each application scenario S_m is modeled as a set of connected tasks represented through the directed task graph $G_m = (\epsilon_m, V_m)$. This task graph has a set of nodes V_m representing tasks and a set of edges ϵ_m representing communication between tasks as shown in the task graph in Figure 2. Each scenario has an execution probability χ_m, an average execution time of τ_m, and a set of tasks $\mathcal{T}_m \subset \mathcal{T}$, where $\mathcal{T}$ is the total set of tasks and $\mathcal{T}_m$ is the set of tasks executed in scenario S_m. The variable $\theta_{i,m}$ indicates the mapping between t_i and S_m such that $\theta_{i,m} = 1$ when $t_i \in \mathcal{T}_m$ and 0 otherwise. For each task $t_i \in \mathcal{T}_m$, there is a computational requirement denoted by γ_i. Communication between the pair of tasks t_i and t_l in S_m is defined by the variable $\varepsilon_{i,l,m}$ where $\varepsilon_{i,l,m} = 1$ if t_i communicates to t_l in S_m and 0 otherwise. The number of transactions between two communicating tasks t_i and t_l in S_m is defined by $N_{trans_{i,l,m}}$. For a certain task t_i, the task probability, ψ_i, is calculated as

$$\psi_i = \sum_{S_m \in \mathcal{S}} \theta_{i,m} \chi_m. \quad (1)$$

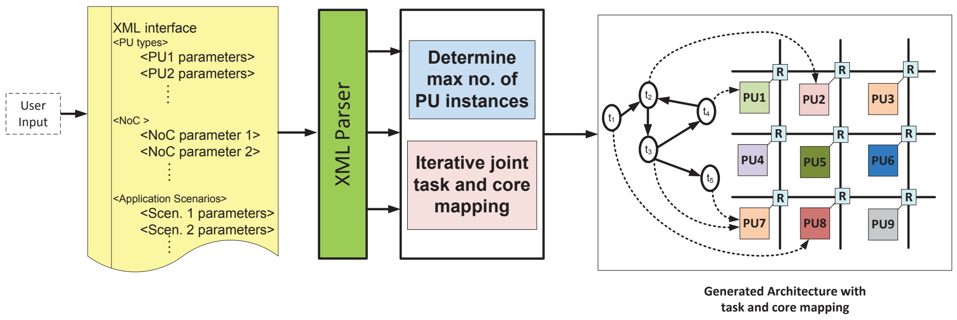


Fig. 1: MPMAP tool flow

B. Architecture model

The model assumes an MPSoC based platform constructed out of different PU types. Each PU type has computational resources denoted as λ_j for j^{th} PU type $P_j \in \mathcal{P}$, where $\mathcal{P}$ is the set of all PU types. It is important to note that even with PUs that are identical in structure, their computational capabilities might be different due to different operating parameters such as supply voltage or operational frequency, where voltage and frequency scaling are commonly utilized for power constrained platforms such as mobile wireless devices. Thus it is reasonable to assume a rich PU space, however to maintain mathematical tractability, we assume that the operational choices such as voltage and frequency settings for each core are preset. Each PU type P_j has a number of instances. The maximum possible number of instances is determined in the design time exploration (DSE) phase.

In general, for heterogeneous PUs, each PU type supports different functional units (FUs), such that some PUs may not be feasible for certain tasks. i.e, one task may require floating point operations which are not supported by certain PUs. Therefore, a feasibility indicator $f_{i,j}$ is defined such that $f_{i,j} = 1$ if task t_i can run on any instance of P_j and 0 otherwise. The communication between these PUs is carried out through network on chip (NoC), which was proven to be efficient in terms of scalability and power consumption [17]. For any instance of the j^{th} PU type, the static power is denoted as σ_j which is independent from the task running over $\hat{p}_{j,k}$, where $\hat{p}_{j,k}$ is the k^{th} instance of the j^{th} PU type. In addition, execution of a task t_i on any $\hat{p}_{j,k}$ is associated with a dynamic power consumption of $\delta_{i,j}$. The execution time of task t_i on any instance $\hat{p}_{j,k}$ is defined by $\tau_{i,j}$. Moreover, we assume that there is a reconfiguration cost associated with each task upon mapping to a certain instance $\hat{p}_{j,k}$. In general, we refer to the reconfiguration power of reconfiguring t_i on any $\hat{p}_{j,k}$ as $\eta_{i,j}$ and the corresponding reconfiguration time with $\tau_{r_{i,j}}$.

The communication between two tasks t_i and t_l mapped on $\hat{p}_{j,k}$ and $\hat{p}_{q,v}$ instances of any PU type is associated with a communication energy cost. This energy is based on the topology and represented as $E_{j,k,q,v}$. In our case, we assume the topology presented in Figure 2, which is a 2D NoC grid connecting different PUs. Each instance $\hat{p}_{j,k}$ is placed in a certain tile defined by a column number $C_{j,k}$ and a row number $R_{j,k}$. The corresponding communication energy per bit between $\hat{p}_{j,k}$ and $\hat{p}_{q,v}$ is represented as $E_{j,k,q,v}$ and it depends on the manhattan distance $MD^{q,v}_{j,k}$ which is the minimum path length between $\hat{p}_{j,k}$ and $\hat{p}_{q,v}$ instances, and is defined as $|C_{j,k} - C_{q,v}| + |R_{j,k} - R_{q,v}|$. The corresponding value of $E_{j,k,q,v}$ is divided into the energy consumed in the link (E_{Lbit}), the energy in the routers (E_{Rbit}), and the energy between the router and the local module (E_{Cbit}) as depicted by (2). The total communication energy is the communication energy per bit times the communication volume between t_i and t_l as indicated by $N_{trans_{i,l,m}}.E_{j,k,q,v}$, such that

$$E_{j,k,q,v} = n_{hops}E_{Rbit} + 2E_{Cbit} + (n_{hops} - 1)E_{Lbit}, \quad (2)$$

where the n_{hops} is the number of routers that the bit passes through, and it is defined as $MD^{q,v}_{j,k} + 1$. To account for the routing through paths our than the shortest distance, a communication overhead factor is defined, and is denoted by C^o. The binary variables $L^c_{j,k}$ and $L^r_{j,k}$ represent the placement of $\hat{p}_{j,k}$ to the c^{th} column and r^{th} row respectively such that $L^{c(r)}_{j,k} = 1$ if $\hat{p}_{j,k}$ is placed in the $c^{th}(r^{th})$ column (row) and 0 otherwise. The overall application and architecture parameters are summarized in Table I.

IV. Problem Formulation

This paper considers joint task and core mapping at the design time phase of heterogeneous MPSoC. For the given set of scenarios $\mathcal{S}$, and a given set of PU types $\mathcal{P}$, the target is to: Choose the appropriate PU instances of each PU type, then find the mapping between different tasks and different PU instances to minimize the overall power consumption including static, dynamic, communication and reconfiguration power. The binding process is accompanied with determination of the number of utilized instances of each PU type. A task t_i is related to the PU $\hat{p}_{j,k}$ through task utilization $u_{i,j} = \frac{\gamma_i}{\lambda_j}$ which defines the portion of PU resources utilized by t_i. In addition, a feasibility indicator is used that indicates whether a task t_i is feasible to run on any instance of P_j. In general, for the mapping of t_i on $\hat{p}_{j,k}$ to be feasible, $u_{i,j} \leq 1$. The variable $M_{i,j,k}$ defines the mapping between t_i and $\hat{p}_{j,k}$ such that $M_{i,j,k} = 1$ if t_i is mapped on $\hat{p}_{j,k}$, 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 $t_i \in \mathcal{T}_m$ is mapped to $\hat{p}_{j,k}$ and 0 otherwise. To formulate a tractable problem, we assume the following:

1) The application scenarios to be run on the architecture are known a priori.

TABLE I: System parameters

Param.	Definition	Param.	Definition
$\mathcal{S}$	Set of scenarios	λ_j	Computational resources of P_j
$\mathcal{T}$	Total set of tasks	$f_{i,j}$	Feasibility indicator of running t_i on any P_j
$\mathcal{P}$	Total set of PU types	$\gamma_{i,j}$	Computational demand of running t_i on P_j
S_m	m^{th} scenario	σ_j	Static power of P_j
χ_m	Execution probability of S_m	$\delta_{i,j}$	Dynamic power of t_i on P_j
$\mathcal{T}_m$	Set of tasks in S_m	$u_{i,j}$	Utilization of t_i on P_j
t_i	i^{th} task	$\eta_{i,j}$	Reconfiguration power of running t_i on P_j
$\theta_{i,m}$	Variable indicates that $t_i \in T_m$	$r_{i,j}$	Average Reconfiguration ratio when running t_i on P_j
τ_m	Execution time of m^{th} scenario	$\hat{p}_{j,k}$	k^{th} instance of P_j
$m_{i,j}$	Activity factor of t_i when running on P_j	$C_{j,k}$	Column number on which $\hat{p}_{j,k}$ is located
$\varepsilon_{i,l,m}$	Variable indicates that t_i communicates to t_l in S_m	$R_{j,k}$	Row number on which $\hat{p}_{j,k}$ is located
$N_{trans_{i,l,m}}$	Number of transactions between t_i and t_l in S_m	$\tau_{i,j}$	Execution time of task t_i on P_j

2) Each task is mapped completely on one and only one PU instance. However, each instance can host more than one task simultaneously.
3) There is at least one feasible PU type to run each task.
4) The probabilities are known a priori. This can be estimated based on historical measurements.
5) The values of static, dynamic, reconfiguration, and communication costs (power or time) of different tasks on different PU instances is obtained through profiling tools.
6) There is no communication cost between any two communicating tasks running on the same PU.

The dynamic power consumption is highly affected by the task activity factor, which depends on the task mapping. The average activity factor associated with mapping a task t_i to any $\hat{p}_{j,k}$ is defined by $m_{i,j}$, and it is computed as:

$$m_{i,j} = \sum_{S_m \in \mathcal{S}} \chi_m \theta_{i,m} \tau_{i,j} \tau_m^{-1} \tag{3}$$

Similarly, we define an average reconfiguration factor, $r_{i,j}$, upon mapping t_i to $\hat{p}_{j,k}$. This average reconfiguration is computed as:

$$r_{i,j} = \sum_{S_m \in \mathcal{S}} \theta_{i,m} \tau_{r_{i,j}} \tau_m^{-1} \tag{4}$$

The reconfiguration process from a scenario S_a to S_m is associated with reconfiguration of tasks $t_i \in \mathcal{T}_m$, and $t_i \notin \mathcal{T}_a$. In addition, we assume equal probability of reconfiguration from one scenario to another scenario, so that it equals $\frac{1}{N_s}$, where N_s is the number of scenarios. Therefore, the reconfiguration power associated with reconfiguration from S_m to S_a is $\sum\limits_{t_i \in \mathcal{T}} \theta_{i,m}(1-\theta_{i,a})\eta_{i,j} M_{i,j,k}$.

In addition, we define an average communication rate, $ACR^{i,l}$, between any two tasks t_i and t_l that represents the average value of communication volume per scenario execution time, and is calculated as:

$$ACR^{i,l} = \sum_{S_m \in \mathcal{S}} \chi_m \theta_{i,m} \theta_{l,m} \varepsilon_{i,l,m} N_{trans_{i,l,m}} \tau_m^{-1} \tag{5}$$

The average power consumption among all scenarios is derived as in (6), where the four terms represent average static power, average dynamic power, average reconfiguration power, and average communication power respectively.

$$\begin{aligned} P_{avg} &= \sum_{S_m \in \mathcal{S}} \sum_{\hat{p}_{j,k} \in \mathcal{P}} \chi_m \sigma_j Z_{m,j,k} \\ &+ \sum_{t_i \in \mathcal{T}} \sum_{\hat{p}_{j,k} \in \mathcal{P}} m_{i,j} \delta_{i,j} M_{i,j,k} \\ &+ \sum_{t_i \in \mathcal{T}} \sum_{\hat{p}_{j,k} \in \mathcal{P}} \frac{1}{N_s} r_{i,j} \left(1-\psi_i\right) \eta_{i,j} M_{i,j,k} \\ &+ \sum_{t_i, t_l \in \mathcal{T}} \sum_{\hat{p}_{j,k}, \hat{p}_{q,v} \in \mathcal{P}} M_{i,j,k} C^o ACR^{i,l} E_{j,k,q,v} M_{l,q,v} \end{aligned} \tag{6}$$

The target is to find the optimal task mapping $M_{i,j,k}$ as well as the optimal core mapping $L^c_{j,k}$, $L^r_{j,k}$ with the target of minimizing the overall power consumption as depicted in the equation set (7).

$$\min \; P_{avg} \tag{7a}$$
$$\text{s.t.} \sum_{\hat{p}_{j,k} \in \mathcal{P}} f_{i,j} M_{i,j,k} = 1 \; \forall \; t_i \in \mathcal{T}_m, \; S_m \in \mathcal{S} \tag{7b}$$
$$\sum_{t_i \in \mathcal{T}_m} u_{i,j} M_{i,j,k} \leq Z_{m,j,k} \quad \forall \; S_m \in S, \; \hat{p}_{j,k} \in \mathcal{P} \tag{7c}$$
$$\sum_{\hat{p}_{j,k} \in \mathcal{P}} L^c_{j,k}.L^r_{j,k} \leq 1 \quad \forall \; c \in \mathcal{C}, \; r \in \mathcal{R} \tag{7d}$$
$$\sum_{c \in \mathcal{C}} \sum_{r \in \mathcal{R}} L^c_{j,k}.L^r_{j,k} = 1 \quad \forall \; \hat{p}_{j,k} \in \mathcal{P} \tag{7e}$$
$$M_{i,j,k} = \{0,1\} \quad \forall \; t_i \in \mathcal{T}, \hat{p}_{j,k} \in \mathcal{P} \tag{7f}$$
$$Z_{m,j,k} = \{0,1\} \quad \forall \; S_m \in \mathcal{S}, \hat{p}_{j,k} \in \mathcal{P} \tag{7g}$$
$$L^c_{j,k} = \{0,1\} \quad \forall \; c \in \mathcal{C}, \hat{p}_{j,k} \in \mathcal{P} \tag{7h}$$
$$L^r_{j,k} = \{0,1\} \quad \forall \; r \in \mathcal{R}, \hat{p}_{j,k} \in \mathcal{P} \tag{7i}$$

The constraint in equation (7b) guarantees that each task is mapped to one and only one feasible PU instance. The constraint in (7c) ensures that the utilization condition is satisfied for each PU instance in each scenario. The constraint (7d) states that each position defined by a row number (r) and a column number (c) in the NoC grid has at most one allocated processing unit, where $\mathcal{R}$ and $\mathcal{C}$ are the sets of total rows and columns in the NoC grid respectively. Additionally, the constraint (7e) guarantees that each PU is allocated to one and only one position in the grid.

V. Proposed heuristic algorithm

We propose an iterative heuristic solution to minimize the average power consumption. The proposed solution has two parts: Initial, and iterative solutions defined by algorithms 1 and 2 respectively. Each one consists of task mapping, followed by core (re)mapping.

A. Static energy aware task mapping

The static power of any PU is task independent, so there is no direct linear cost function that relates the mapping variable $M_{i,j,k}$ to the average static power dissipation. An initial relaxed cost function, shown in step 5, is used to estimate the average power consumption of each task on any PU. This initial cost function considers only the effect of static, dynamic, and reconfiguration power costs. There is no corresponding available communication cost prior to mapping. The set of tasks with the minimum cost function on P_j are included in $\mathcal{T}_j$ and all the tasks in $\mathcal{T}_j$ 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 PU instances with lower static power consumption, and contributes accordingly to the total power saving. As shown in step 8, the PU types are sorted in ascending order according to the static power consumption. The power cost is then computed for each $t_i \in \mathcal{T}_j$ on several PU instances as depicted in steps 14-17 . In step 19, the task t_i is mapped to $\hat{p}_{\breve{j},\breve{k}}$ with the minimum total estimated cost. After mapping t_i to $\hat{p}_{\breve{j},\breve{k}}$, the total utilization $U_{m,\breve{j},\breve{k}}$ of $\hat{p}_{\breve{j},\breve{k}}$ in each scenario S_m that includes t_i is updated as shown in step 20, and mapping variable is updated as in step 21. Afterwards, the core mapping algorithm defined by algorithm 3 is executed based on the available communication costs.

Algorithm 1 Initial task allocation

```
1:  $\mathcal{T}_j \leftarrow \phi \;\forall\; P_j \in \mathcal{P}$
2:  $Z_{m,j,k} = 0 \;\forall\; S_m \in \mathcal{S},\; P_j \in \mathcal{P}$
3:  $U_{m,j,k} = 0 \;\forall\; S_m \in \mathcal{S},\; P_j \in \mathcal{P}$
4:  for each $t_i \in \mathcal{T}$ do
5:      $P_j \leftarrow \arg\min_{P_j \in \mathcal{P}} \{\psi_i u_{i,j}\sigma_j + m_{i,j}\delta_{i,j} + (1-\psi_i) r_{i,j}\eta_{i,j}/N_s\}$ such that $f_{i,j} = 1$ and $u_{i,j} \le 1$
6:      $T_j \leftarrow T_j \cup \{t_i\}$
7:  end for
8:  Sort $P_j \in \mathcal{P}$ in ascending order according to static power
9:  for each $P_j \in \mathcal{P}$ do
10:     Sort all $t_i \in \mathcal{T}_j$ in descending order according to the utilization $u_{i,j}$
11:     for each $t_i \in \mathcal{T}_j$ do
12:         for each $\hat{p}_{j',k'} \in \mathcal{P}$ do
13:             Let $S_{t_i}$ be the set of scenarios which host task $t_i$
14:             $S\_cost_{i,j',k'} = \sum_{S_m \in \mathcal{S}} \chi_m Z_{m,j',k'} \sigma_{j'}$
15:             $D\_cost_{i,j',k'} = m_{i,j'}\delta_{i,j'}$
16:             $R\_cost_{i,j',k'} = (1-\psi_i) r_{i,j'}\eta_{i,j'}/N_s$
17:             $Cost_{i,j',k'} = S\_cost_{i,j',k'} + D\_cost_{i,j',k'} + R\_cost_{i,j',k'}$
18:         end for
19:         Assign $t_i$ to $\hat{p}_{\breve{j},\breve{k}}$ such that:
            $\hat{p}_{\breve{j},\breve{k}} \leftarrow \arg\min_{\hat{p}_{j',k'}} \{Cost_{i,j',k'}\}$,
            $U_{m,\breve{j},\breve{k}} + u_{i,\breve{j}} \le 1 \;\forall\; S_m \in S_{t_i}$ and $f_{i,\breve{j}} = 1$
20:         $U_{m,\breve{j},\breve{k}} = U_{m,\breve{j},\breve{k}} + u_{i,\breve{j}} \;\forall\; S_m \in S_{t_i}$
21:         $Z_{m,\breve{j},\breve{k}} = 1 \;\forall\; S_m \in S_{t_i}$ , $M_{i,\breve{j},\breve{k}} = 1$
22:     end for
23: end for
24: Core_mapping_algorithm () ;
```

B. Iterative task remapping

After the initial mapping, an iterative solution is used to remap tasks to reduce the average power consumption as presented in algorithm 2. The set of PU instances that host tasks, $\mathcal{P}^o$, are defined after initial mapping. At each iteration, the set of PU instances in $\mathcal{P}^o$ are sorted in ascending order according to maximum utilization as shown in step 2. Accordingly, remapping starts first to tasks hosted by low utilized PUs in order to increase the chance to free up the low utilized PU instances. A new cost is calculated for each task t_i on each $\hat{p}_{j',k'}$ based on static, dynamic, reconfiguration and communication cost as shown in step 7. The communication cost is estimated based on the recent task and core mapping. As depicted in step 9, the task t_i is assigned to the PU $\hat{p}_{\breve{j},\breve{k}}$ with the minimum total energy cost and satisfies the utilization constraint for all scenarios S_{t_i} hosting t_i. The resulting average energy consumption is reduced with the increase in the number of iterations and the performance approaches the solutions obtained via optimal solvers.

Algorithm 2 Iterative task allocation

```
1:  for $Niter = 2 : no\_iter$ do
2:      Sort occupied PU instances $\mathcal{P}^o$ in ascending order based on max utilization
3:      for each $\hat{p}_{j_p,k_p} \in \mathcal{P}^o$ do
4:          for each $t_i$ hosted by $\hat{p}_{j_p,k_p}$ do
5:              Let $S_{t_i}$ is the set of scenarios associated with $t_i$
6:              for each $\hat{p}_{j',k'} \in \mathcal{P}$ do
7:                  $Est\_Cost_{i,j',k'} = S\_cost_{i,j',k'} + D\_cost_{i,j',k'} + R\_cost_{i,j',k'} + C\_cost_{i,j',k'}$
8:              end for
9:              Assign $t_i$ to $\hat{p}_{\breve{j},\breve{k}}$ such that:
                $\hat{p}_{\breve{j},\breve{k}} \leftarrow \arg\min_{\hat{p}_{j',k'}} \{Est\_Cost_{i,j',k'}\}$ ,
                $U_{m,\breve{j},\breve{k}} + u_{i,j} \le 1 \;\forall\; S_m \in S_{t_i}$ and $f_{i,\breve{j}} = 1$
10:             $U_{m,j_p,k_p} = U_{m,j_p,k_p} - u_{i,j_p} \;\forall\; S_m \in S_{t_i}$
11:             $Z_{m,j_p,k_p} = 0 \;\forall\; S_m \in S_{t_i}$ such that $\hat{p}_{j_p,k_p}$ does not host any other $t_l \in \mathcal{T}$ with $\theta_{l,m} = 1$
12:             $M_{i,j_p,k_p} = 0$
13:             $U_{m,\breve{j},\breve{k}} = U_{m,\breve{j},\breve{k}} + u_{i,\breve{j}} \;\forall\; S_m \in S_{t_i}$
14:             $Z_{m,\breve{j},\breve{k}} = 1 \;\forall\; S_m \in S_{t_i}$ , $M_{i,\breve{j},\breve{k}} = 1$
15:         end for
16:     end for
17:     Core_mapping_algorithm ();
18: end for
```

C. Core Mapping

After each mapping iteration, the core mapping heuristic in algorithm 3 is used to locate each utilized PU instance in a specific tile in the 2D NoC such that the communication power 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 are defined by P_p and P_u respectively. The core mapping is determined based on the average communication rates between the different instances. The algorithm tends to map the instances with higher communication rates as close as possible. Consider N^o as the total number of utilized instances resulting from the DSE in the mapping algorithm, the upper limit of the grid size is $\lceil\sqrt{N^o}\rceil$ x $\lceil\sqrt{N^o}\rceil$. As presented in step 4, the core mapping heuristic chooses the PU instance with the highest average 2-way communication rate with other instances as the first one to be mapped. It is indicated as $\hat{p}_{j_m,k_m}$, and it is

TABLE II: Description of realistic benchmarks

Scen.	Description	Tasks/nodes/edges	Prob.
S_0	A_1	7 / 7 / 6	0.118
S_1	A_2	13 / 13 / 12	0.118
S_2	A_3	4 / 8 / 9	0.092
S_3	A_4	7 / 12 /14	0.092
S_4	A_1, A_3	11 / 15 / 15	0.026
S_5	A_1, A_4	14 / 19 / 20	0.038
S_6	A_2, A_3	17 / 21 / 21	0.026
S_7	A_2, A_4	20 / 25 / 26	0.026
S_8	A_1, A_3, A_4	18 / 28 / 29	0.022
S_9	A_2, A_3, A_4	24 / 33 / 35	0.022
S_{10}	A_1, A_2, A_3, A_4	31 / 40 / 42	0.420

TABLE III: Description of the used PU types

PU type	Vdd (V)	Tech. (nm)	Area (mm^2)	Freq. (MHZ)	Stat. pow. (mW)	Dyn. pow. (mW/MHZ)
OpenRISC 1200	1.0	65	0.052	250	0.263	0.025
ARM-Cortex A9	1.0	65	5.7	830	24.6	0.707
UltraSPARC-T1	1.0	65	7.57	610	51.1	0.633
Large Rec. fabric	1.0	65	40.8	100	312	0.0011/CLB
Small Rec. fabric	1.0	65	5.1	100	39.06	0.0011/CLB
Turbo dec. core	1.0	65	0.938	100	9.1	0.608

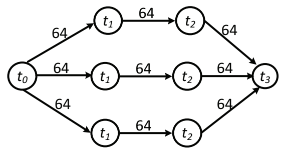
mapped in the middle of the hypothetical maximum size grid as presented in step 5, where C_{j_m,k_m} and R_{j_m,k_m} indicate the column and row positions of $\hat{p}_{j_m,k_m}$ respectively. Then, other instances are mapped iteratively in descending order according to the average communication rates with the currently mapped instances. The position of the selected instance is determined such as it minimizes the total communication cost with the mapped instances as indicated in step 9. The algorithm repeats this process until all the instances are mapped to the grid. Whenever a new instance is mapped, it is inserted in the set of the mapped instances P_p and removed from the set of unmapped instances P_u as shown in step 10.

Algorithm 3 Core Mapping

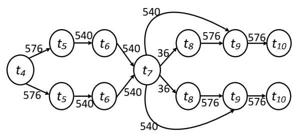
```
1: $P^o \leftarrow$ Set of occupied instances that host tasks
2: $P_p = \phi$ , $P_u = P_o$
3: $Navg\_out_{j,k} \leftarrow$ Number of average 2-way comm. rates with each $\hat{p}_{j,k}$
4: $\hat{p}_{j_m,k_m} \leftarrow arg \max_{P_j \in P} \{Navg\_out_{j,k}\}$
5: $C_{j_m,k_m} = \lceil\sqrt{N^o}\rceil/2$ , $R_{j_m,k_m} = \lceil\sqrt{N^o}\rceil/2$
6: $P_p = P_p \bigcup p_{j_m,k_m}$ , $P_u = P_u \backslash p_{j_m,k_m}$
7: while $P_u \neq \phi$ do
8:     $\hat{p}_{j_m,k_m} \leftarrow arg \max_{P_j \in P_u}$ {no. of 2-way trans. with instances in $P_p$}
9:     $R_{j_m,k_m} = r$, and $C_{j_m,k_m} = c$ such that the $(r,c)^{th}$ location
       minimizes the comm. energy cost with the recently placed instances.
10:    $P_p = P_p \bigcup p_{j_m,k_m}$ , $P_u = P_u \backslash p_{j_m,k_m}$
11: end while
```

VI. Performance Results

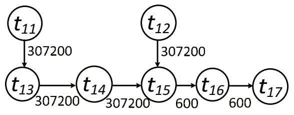
The model in section IV is populated over a hypothetical example of a platform supporting heterogeneous wireless networks and running two wireless (A1: WCDMA RX [18] , A2: LTE RX [19]) and two multimedia (A3: JPEG encoder [20] , A4: mp3 decoder [21]) examples, constructing 11 execution scenarios as shown in Table II. The table also illustrates the scenario probabilities. This examples assumes the library of available PU types as presented in Table III.



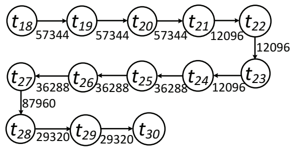
(a) JPEG encoder task graph



(b) mp3 decoder task graph



(c) WCDMA receiver task graph



(d) LTE receiver task graph

t_0	RGB Conversion	t_{10}	Subband synthesis	t_{21}	OFDM Demod
t_1	IDCT	t_{11}	WCDMA RX filter	t_{22}	Pre-decoding
t_2	Quantize	t_{12}	RX PN generator	t_{23}	Layer Demapper
t_3	Huffman encoding	t_{13}	Path search	t_{24}	Modulation demapper
t_4	Huffman decoding	t_{14}	Descrambling	t_{25}	Descrambling
t_5	Requantizer	t_{16}	Deinterleaving	t_{26}	De-concatenation
t_6	Reorder	t_{17}	Viterbi deocder	t_{27}	Rate unmatching
t_7	Stereo	t_{18}	Coarse timing and freq. offset	t_{28}	Turbo decoding
t_8	Alias	t_{19}	Primary sync. and fine timing	t_{29}	Desegmentation
t_9	Hybrid synthesis	t_{20}	Secondary synchronization	t_{30}	CRC Check

(e) List of the individual tasks

Fig. 3: Task graph of different applications

Profiling of different scenarios on different PU types was held through different profiling tools such as OpenRISC Or1Ksim, GEM5 [22], McPAT [23], and different data are obtained from [24]–[29]. The performance of the proposed heuristic is compared to some of the existing heuristics. To be able to compare it to optimal solvers, the task mapping only is considered, assuming predefined core mapping, and the heuristic performance is compared to the IBM CPLEX solver [30], a simulated annealing heuristic based on [6], a baseline heuristic that considers only static and dynamic power based on [5]. Then, the joint mapping improvement is illustrated with respect to the task mapping-only. Table IV illustrates that the heuristic performance is close to the CPLEX optimal solver with orders of magnitude speedup in execution time. The proposed heuristic is efficient with higher problem sizes where using optimal solvers is infeasible. The resulting MPMAP output is shown in Figure 4, where the synthesized architecture has 7 utilized PU instances. The figure also shows the corresponding task mapping to cores and core mapping to tiles.

TABLE IV: Performance of different mapping approaches

	Power consumption	Algorithm exec. time
Proposed heur. 4 iter	**680.04 mW**	**24.05 ms**
Stat. and dyn heur. 4 iter	819.59 mW	23.18 ms
SA-based heur. 100 iter	1305.39 mW	120.6 ms
SA-based heur. 1000 iter	894.746 mW	1.22 sec.
CPLEX (1 min.)	829.05 mW	1 min.
CPLEX (10 min.)	811.93 mW	10 min.
CPLEX (1 hr.)	683.08 mW	1 hour
Joint task and core mapping	**648.35 mW**	**26.84 ms**

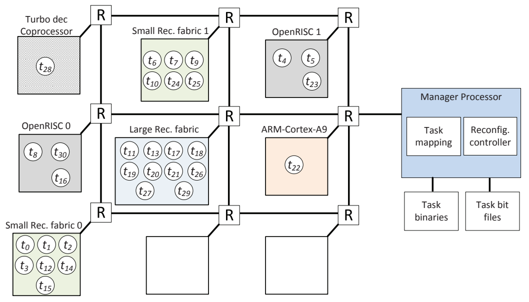


Fig. 4: Generated platform with task mapping

VII. Conclusion

The paper proposes a high level synthesis tool that performs power aware joint task and core mapping considering static, dynamic, reconfiguration and communication powers. The task mapping heuristic provides comparable performance and extensive speedup in execution time with respect to the CPLEX optimal solver. Additional power saving is achieved through joint task and core mapping.

References

[1] W. Wolf, A. Jerraya, and G. Martin, "Multiprocessor System-on-Chip (MPSoC) Technology," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 10, pp. 1701 –1713, oct. 2008.

[2] F. Wronski, E. W. Brião, and F. R. Wagner, "Evaluating Energy-Aware Task Allocation Strategies for MPSoCs," *IFIP International Federation for Information Processing*, vol. 225, pp. 215–224, 2006.

[3] R. Xu, R. Melhem, and D. Mosse, "Energy-Aware Scheduling for Streaming Applications on Chip Multiprocessors," in *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, dec. 2007, pp. 25 –38.

[4] L. Thiele, L. Schor, H. Yang, and I. Bacivarov, "Thermal-aware system analysis and software synthesis for embedded multi-processors," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, 2011, pp. 268–273.

[5] A. Schranzhofer, J.-J. Chen, and L. Thiele, "Dynamic Power-Aware Mapping of Applications onto Heterogeneous MPSoC Platforms," *Industrial Informatics, IEEE Transactions on*, vol. 6, no. 4, pp. 692 –707, nov. 2010.

[6] C. Jueping, Y. Lei, H. Gang, H. Yue, W. Shaoli, and L. Zan, "Communication- and energy-aware mapping on homogeneous NoC architecture with shared memory," in *Solid-State and Integrated Circuit Technology (ICSICT), 2010 10th IEEE International Conference on*, nov. 2010, pp. 290 –292.

[7] S. Manolache, P. Eles, and Z. Peng, "Task mapping and priority assignment for soft real-time applications under deadline miss ratio constraints," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 2, pp. 19:1–19:35, Jan. 2008.

[8] J. Choi, H. Oh, S. Kim, and S. Ha, "Executing synchronous dataflow graphs on a SPM-based multicore architecture," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, 2012, pp. 664–671.

[9] A. Bender, "MILP based task mapping for heterogeneous multiprocessor systems," in *Design Automation Conference, 1996, with EURO-VHDL '96 and Exhibition, Proceedings EURO-DAC '96, European*, 1996, pp. 190–197.

[10] P. Ghosh, A. Sen, and A. Hall, "Energy efficient application mapping to NoC processing elements operating at multiple voltage levels," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, 2009, pp. 80–85.

[11] J. Huang, C. Buckl, A. Raabe, and A. Knoll, "Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems," in *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, 2011, pp. 447–454.

[12] G. Chen, F. Li, S. Son, and M. Kandemir, "Application mapping for chip multiprocessors," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, 2008, pp. 620–625.

[13] S. Kaushik, A. Singh, W. Jigang, and T. Srikanthan, "Run-Time Computation and Communication Aware Mapping Heuristic for NoC-Based Heterogeneous MPSoC Platforms," in *Parallel Architectures, Algorithms and Programming (PAAP), 2011 Fourth International Symposium on*, 2011, pp. 203–207.

[14] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," *IEEE/ACM Design Automation Conference (DAC)*, 2013.

[15] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 4, pp. 551–562, 2005.

[16] C. A. M. Marcon, E. Moreno, N. L. V. Calazans, and F. Moraes, "Comparison of network-on-chip mapping algorithms targeting low energy consumption," *Computers Digital Techniques, IET*, vol. 2, no. 6, pp. 471–482, 2008.

[17] J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: a scalable, communication-centric embedded system design paradigm," in *VLSI Design, 2004. Proceedings. 17th International Conference on*, 2004, pp. 845–851.

[18] "SDRG WCDMA Benchmark Suite." [Online]. Available: http://web.eecs.umich.edu/ sdrg/benchmarks/wcdmaBench_1.1.tar.gz

[19] "3GPP-LTE open source implementation." [Online]. Available: http://sourceforge.net/p/openlte/home/Home/

[20] "IJG JPEG Benchmark Suite." [Online]. Available: http://www.ijg.org/files/jpegsrc.v8d.tar.gz

[21] "mp3 decoder Benchmark Suite." [Online]. Available: http://www.es.ele.tue.nl/sdf3/download/benchmarks.php

[22] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[23] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 469–480.

[24] M. Vestias and H. Neto, "Co-synthesis of a configurable soc platform based on a network on chip architecture," in *Design Automation, 2006. Asia and South Pacific Conference on*, jan. 2006, p. 6 pp.

[25] H. Hedberg, T. Lenart, and H. Svensson, "A complete mp3 decoder on a chip," in *Microelectronic Systems Education, 2005. (MSE '05). Proceedings. 2005 IEEE International Conference on*, june 2005, pp. 103 – 104.

[26] E. Grayver, J.-F. Frigon, A. Eltawil, A. Tarighat, K. Shoarinejad, A. Abbasfar, D. Cabric, and B. Daneshrad, "Design and vlsi implementation for a wcdma multipath searcher," *Vehicular Technology, IEEE Transactions on*, vol. 54, no. 3, pp. 889 – 902, may 2005.

[27] R. Asghar and D. Liu, "Dual standard re-configurable hardware interleaver for turbo decoding," in *Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on*, may 2008, pp. 768 – 772.

[28] M. Shafique, L. Bauer, and J. Henkel, "REMiS: Run-time energy minimization scheme in a reconfigurable processor with dynamic power-gated instruction set," in *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, nov. 2009, pp. 55 –62.

[29] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "A 390Mb/s 3.57mm^2 3GPP-LTE turbo decoder ASIC in 0.13 m CMOS," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, feb. 2010, pp. 274 –275.

[30] "IBM ILOG CPLEX Optimization Studio, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/."