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 χ_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 τ_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 $\mathcal{T}$, and the set $\mathcal{T}_m$ is the set of tasks executed in scenario S_m.

The system comprises a set of PU types $\mathcal{P}$, where the jth PU type, P_j, can have multiple instances such that $\hat{p}_{j,k}$ is the kth instance of P_j. The maximum possible number of instances F_j of the jth PU type is determined at design time based on the task requirements. The static power of P_j is σ_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 ith task, t_i, on any instance of P_j are $\delta_{i,j}$ and $\eta_{i,j}$, while the corresponding execution and reconfiguration times are $\tau_{i,j}$ and $\tau_{r_{i,j}}$. Each task t_i has an execution probability of ψ_i, average number of executions of κ_i^{av}, and reconfiguration probability of r_i. The value of r_i depends on the transition probabilities among different execution scenarios. The feasibility indicator $f_{i,j}$ is used to identify if it is

Manuscript received February 10, 2013; revised July 10, 2013 and October 29, 2013; accepted February 6, 2014. Date of publication March 18, 2014; date of current version January 30, 2015. This work was supported in part by the National Science Foundation under Grant ECCS-0955157 and in part by the Qatar National Research Fund under Grant 4-1119-2-427.

A. M. A. Hussien and A. M. Eltawil are with the Center of Embedded Computer Systems, Center for Pervasive Communications and Computing, University of California at Irvine, Irvine, CA 92697 USA (e-mail: ahussien@uci.edu; aeltawil@uci.edu).

R. Amin and J. Martin are with the Network Laboratory, School of Computing, Clemson University, Clemson, SC 29634 USA (e-mail: ramin@clemson.edu; jim.martin@cs.clemson.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2309013

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 $\hat{p}_{j,k}$ and t_l mapped to $\hat{p}_{q,v}$ 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 $\hat{p}_{j,k}$ with a width $w_{j,k}$, and height $h_{j,k}$, the core location is defined by the x–y coordinates such that $(X^{\min}_{j,k}, Y^{\min}_{j,k})$ are the coordinates of the lower left side, and $(X^{\max}_{j,k}, Y^{\max}_{j,k})$ 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 $\hat{p}_{j,k}$ and $\hat{p}_{q,v}$ are represented as $\Delta X^{q,v}_{j,k} = X^{\max}_{j,k} - X^{\max}_{q,v}$, and $\Delta Y^{q,v}_{j,k} = Y^{\max}_{j,k} - Y^{\max}_{q,v}$. E^{bit}_l 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 $\hat{p}_{j,k}$ 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 $\hat{p}_{j,k}$ 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. The binary variables $\mathcal{X}^{q,v}_{j,k}$ and $\mathcal{Y}^{q,v}_{j,k}$ determines the relative position of PU instances $\hat{p}_{j,k}$, and $\hat{p}_{q,v}$. The variable $\mathcal{X}^{q,v}_{j,k} = 1$ if $X^{\min}_{j,k} \geq X^{\max}_{q,v}$, and 0 otherwise. This represents that the PU instance $\hat{p}_{j,k}$ is completely located to the right of $\hat{p}_{q,v}$. Similarly, $\mathcal{Y}^{q,v}_{j,k} = 1$ if the PU instance $\hat{p}_{j,k}$ is completely located to the top of $\hat{p}_{q,v}$.

The average energy consumption among all scenarios is derived as

$$\begin{aligned} E_{\text{avg}} = &\sum_{S_m \in \mathcal{S}} \sum_{\hat{p}_{j,k} \in \mathcal{P}} \chi_m \tau_m \sigma_j Z_{m,j,k} \\ &+ \sum_{t_i \in \mathcal{T}} \sum_{\hat{p}_{j,k} \in \mathcal{P}} \{\kappa^{av}_i \tau_{i,j} \delta_{i,j} + r_i \eta_{i,j} \tau_{r_{i,j}}\} M_{i,j,k} \\ &+ \sum_{t_i, t_l \in \mathcal{T}} \sum_{\hat{p}_{j,k}, \hat{p}_{q,v} \in \mathcal{P}} \bar{Q}_{i,l} E^{\text{bit}}_l \Big\{ WX^{+\,l,q,v}_{\;\;i,j,k} + WX^{-\,l,q,v}_{\;\;i,j,k} \\ &\qquad\qquad + WY^{+\,l,q,v}_{\;\;i,j,k} + WY^{-\,l,q,v}_{\;\;i,j,k} \Big\} \end{aligned} \tag{1}$$

where $\bar{Q}_{i,l} = \sum_{S_m \in \mathcal{S}} \chi_m Q^m_{i,l} / \omega^2_{i,l,m}$, $Q^m_{i,l}$ is the communication volume between t_i and t_l in S_m, and $\omega_{i,l,m}$ is a weighting factor representing the latency constraint in hops [9] between t_i and t_l 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 $(X^{\min}_{j,k}, Y^{\min}_{j,k})$ 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

Algorithm 1 Initial Task Allocation

```
1: $\mathcal{T}_j \leftarrow \phi \;\forall\; P_j \in \mathcal{P}$
2: $Z_{m,j,k} = 0 \;\forall\; S_m \in \mathcal{S},\; \hat{p}_{j,k} \in \mathcal{P}$
3: $U_{m,j,k} = 0 \;\forall\; S_m \in \mathcal{S},\; \hat{p}_{j,k} \in \mathcal{P}$
4: for each $t_i \in \mathcal{T}$ do
5:     Determine $P_j$ type which minimizes the expected energy consumption
       based on static, dynamic, and reconfiguration energy costs only such
       that $f_{i,j} = 1$ and $u_{i,j} \leq 1$.
6:     $P_j \leftarrow \arg\min_{P_j \in \mathcal{P}} \{u_{i,j}\zeta_i\sigma_j + \kappa^{av}_i \tau_{i,j}\delta_{i,j} + r_i\eta_{i,j}\tau_{r_{i,j}}\}$
7:     $\mathcal{T}_j \leftarrow \mathcal{T}_j \cup \{t_i\}$
8: end for
9: Sort PU types $P_j \in \mathcal{P}$ in ascending order according to static power
   consumption
10: for each $P_j \in \mathcal{P}$ do
11:     Sort all tasks in $\mathcal{T}_j$ in descending order according to the utilization
        $u_{i,j}$
12:     for each $t_i \in \mathcal{T}_j$ do
13:         for each $\hat{p}_{j',k'} \in \mathcal{P}$ do
14:             Let $S_{t_i}$ be the set of scenarios which host task $t_i$
15:             $S\_cost_{i,j',k'} = \sum_{S_m \in S_{t_i}} \chi_m\tau_m Z_{m,j',k'}\sigma_{j'}$
16:             $D\_cost_{i,j',k'} = \kappa^{av}_i \tau_{i,j'}\delta_{i,j'}$
17:             $R\_cost_{i,j',k'} = r_i\eta_{i,j'}\tau_{r_{i,j'}}$
18:             $Cost_{i,j',k'} = S\_cost_{i,j',k'} + D\_cost_{i,j',k'} + R\_cost_{i,j',k'}$
19:         end for
20:         Assign $t_i$ to $\hat{p}_{\check{j},\check{k}}$ such that:
            $\hat{p}_{\check{j},\check{k}} \leftarrow \arg\min_{\hat{p}_{j',k'}} \{Cost_{i,j',k'}\}$,
            $U_{m,\check{j},\check{k}} + u_{i,\check{j}} \leq 1 \;\forall\; S_m \in S_{t_i}$ and $f_{i,\check{j}} = 1$
21:         $U_{m,\check{j},\check{k}} = U_{m,\check{j},\check{k}} + u_{i,\check{j}} \;\forall\; S_m \in S_{t_i}$
22:         $Z_{m,\check{j},\check{k}} = 1 \;\forall\; S_m \in S_{t_i}$ , $M_{i,\check{j},\check{k}} = 1$
23:     end for
24: end for
25: Core_mapping_algorithm () ;
```

used to substitute the multiplication of the two binary and continuous terms, where D is a very large constant

$$\min\ E_{\text{avg}} \tag{2a}$$

$$\text{s.t.} \quad \sum_{P_j \in \mathcal{P}} \sum_{k=1}^{F_j} f_{i,j} M_{i,j,k} = 1 \quad \forall\ t_i \in \mathcal{T}_m, \quad S_m \in \mathcal{S} \tag{2b}$$

$$\sum_{t_i \in \mathcal{T}_m} u_{i,j} M_{i,j,k} \leq Z_{m,j,k} \quad \forall\ S_m \in \mathcal{S}, \quad \hat{p}_{j,k} \in \mathcal{P} \tag{2c}$$

$$\mathcal{X}^{q,v}_{j,k} + \mathcal{X}^{j,k}_{q,v} + \mathcal{Y}^{q,v}_{j,k} + \mathcal{Y}^{j,k}_{q,v} \geq 1 \quad \forall \hat{p}_{j,k}, \quad \hat{p}_{q,v} \in \mathcal{P} \tag{2d}$$

$$\begin{aligned} &WX^{+\,l,q,v}_{\;\;i,j,k} - \Delta X^{+\,q,v}_{\;\;j,k} + D(2 - M_{i,j,k} - M_{l,q,v}) \geq 0 \\ &WX^{-\,l,q,v}_{\;\;i,j,k} - \Delta X^{-\,q,v}_{\;\;j,k} + D(2 - M_{i,j,k} - M_{l,q,v}) \geq 0 \\ &WY^{+\,l,q,v}_{\;\;i,j,k} - \Delta Y^{+\,q,v}_{\;\;j,k} + D(2 - M_{i,j,k} - M_{l,q,v}) \geq 0 \\ &WY^{-\,l,q,v}_{\;\;i,j,k} - \Delta Y^{-\,q,v}_{\;\;j,k} + D(2 - M_{i,j,k} - M_{l,q,v}) \geq 0 \\ &\qquad\qquad \forall\ t_i, t_l \in \mathcal{T} \quad \hat{p}_{j,k}, \hat{p}_{q,v} \in \mathcal{P}. \end{aligned} \tag{2e}$$

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

Algorithm 2 Core (Re)Mapping Algorithm

```
1: $\mathcal{P}^o \leftarrow$ Set of occupied instances that host tasks
2: $P_p = \phi$ ;   $P_u = \mathcal{P}^o$;
3: $R \leftarrow$ Set of feasible positions in X-Y plane
4: $Navg\_out_{j,k} \leftarrow$ Average 2-way comm. cost with each $\hat{p}_{j,k}$
5: $\hat{p}_{j_m,k_m} \leftarrow arg \max_{\hat{p}_{j,k} \in \mathcal{P}} \{Navg\_out_{j,k}\}$
6: $X^{min}_{j_m,k_m} = (0,0)$
7: $r \leftarrow$ Area bounded by $(X^{min}_{j_m,k_m}, Y^{min}_{j_m,k_m})$ and $(X^{max}_{j_m,k_m}, Y^{max}_{j_m,k_m})$
8: $P_p = P_p \bigcup p_{j_m,k_m}$ ;   $P_u = P_u \backslash p_{j_m,k_m}$;   $R = R \backslash r$;
9: while $P_u \neq \phi$ do
10:     $\hat{p}_{j_m,k_m} \leftarrow arg \max_{P_j \in P_u}$ {2-way comm. cost with instances in $P_p$}
11:     $(X^{min}_{j_m,k_m}, Y^{min}_{j_m,k_m}) = (X,Y) \in R$ that minimizes the comm.
        energy cost with the between $\hat{p}_{j_m,k_m}$ and placed instances.
12:     $r \leftarrow$ Area bounded by $(X^{min}_{j_m,k_m}, Y^{min}_{j_m,k_m})$ and $(X^{max}_{j_m,k_m}, Y^{max}_{j_m,k_m})$
13:     $P_p = P_p \bigcup p_{j_m,k_m}$;   $P_u = P_u \backslash p_{j_m,k_m}$;   $R = R \backslash r$;
14: end while
```

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 $\mathcal{T}_j$ 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 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 $t_i \in \mathcal{T}_j$ 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 $\hat{p}_{\check{j},\check{k}}$ with the minimum total estimated cost. After mapping t_i to $\hat{p}_{\check{j},\check{k}}$, the total utilization of $\hat{p}_{\check{j},\check{k}}$ for each scenario S_m that host t_i, denoted by $U_{m,\check{j},\check{k}}$, 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 $\mathcal{P}^o$. 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 $\hat{p}_{j_m,k_m}$, and it is mapped with the left lower side $(X^{\min}_{j_m,k_m}, Y^{\min}_{j_m,k_m})$ at $(0,0)$. As a consequence, the rectangular area bounded by $(X^{\min}_{j_m,k_m}, Y^{\min}_{j_m,k_m})$, and $(X^{\max}_{j_m,k_m}, Y^{\max}_{j_m,k_m})$, 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.

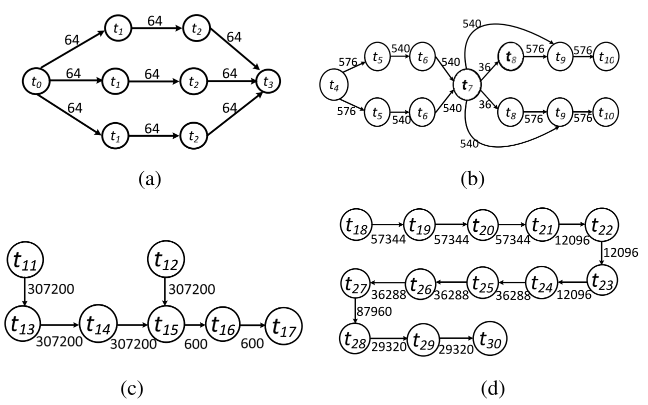
The algorithm complexity is based on the number of tasks (N_t), number of PU types (N_p), number of resulting PU instances (N_i), and number of scenarios (N_s). The complexity of the initial mapping $\mathcal{O}(N_p N_t N_i N_s)$, while the iterative mapping is $\mathcal{O}(N_i^2 N_t N_s)$.

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 A_1, LTE, denoted by A_2, JPEG encoder, denoted by A_3 and MP3 decoding, denoted by A_4. 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,



(e)

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

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

Scen.	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
Desc.	A_1	A_2	A_3	A_4	A_1, A_3 A_3	A_1, A_4 A_4	A_2, A_3 A_3	A_2, A_4 A_4	A_1, A_3, A_4	A_2, A_3, A_4	A_1, A_2, A_3, A_4
Prob.	0.118	0.118	0.092	0.092	0.026	0.038	0.026	0.026	0.022	0.022	0.420

TABLE II
DESCRIPTION OF THE USED PU TYPES

PU type	Vdd (V)	Tech. (nm)	Area (mm^2)	Freq. (MHZ)	Static power (mW)	Dynamic power (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

ARM-Cortex-A9, and Ultra-SPARC-T1 is calculated as the ratio between task execution time and task deadline based on the application throughput. The utilization of FPGA is calculated as a ratio between the task requirements and the available resources in terms of number of CLBs.

The proposed heuristic is applied to the set of the aforementioned scenarios on the hypothetical platform, and performance is evaluated. Performance of the proposed heuristic is compared with the CPLEX solver solution, the SA-based heuristic [23], and a baseline heuristic that considers the static and dynamic powers only [4]. Table III presents the energy consumption and the algorithm execution time of the different algorithms. As shown from the table, the proposed heuristic achieves 23.33% energy saving with respect to the heuristic that considers only static and dynamic power. It achieves 65.11% (51.43%) energy saving with respect to the SA approach running at 100 (1000) iterations. Table III also highlights the heuristic performance with respect to the CPLEX solution upon setting the CPLEX time limit to different values. The proposed heuristic outperforms the CPLEX optimizer when running CPLEX for 1 and 5 min, respectively. The CPLEX outperforms the heuristic performance after running for 1 h, the results show that the overhead of the heuristic performance is 7.1% of the CPLEX solution. The CPLEX optimizer was not able to handle the same problem with a larger number of PU types due to the extensive memory requirements. This confirms that using the proposed heuristic is necessary, especially with higher dimension problem sizes where using the CPLEX optimizer becomes infeasible. The compiled architecture consists of seven PU instances divided as follows: 1) two OpenRISC cores; 2) one ARM-Cortex-A9 core; 3) one coarse reconfigurable fabric; 4) two fine reconfigurable fabrics; and 5) one turbo decoder coprocessor core. The utilized PU instances are mapped to build the NoC-based MPSoC. Fig. 2 shows the generated architecture with the corresponding task mapping and floorplanning.

TABLE III
PERFORMANCE OF DIFFERENT APPROACHES FOR TASK MAPPING ON A PREDEFINED ARCHITECTURE

	Energy consumption	Algorithm exec. time
Proposed heuristic	1494.8 mJ	66.2 ms
CPLEX (1 min.)	2324.8 mJ	1 min.
CPLEX (5 min.)	1594.9 mJ	5 min.
CPLEX (1 hr.)	1392.4 mJ	1 hour
Stat. and dyn. only	1949.7 mJ	46.7 ms
SA-based heur. (100 Iter)	4283.8 mJ	1.3 sec
SA-based heur. (1000 Iter)	3077.38 mJ	10.7 sec

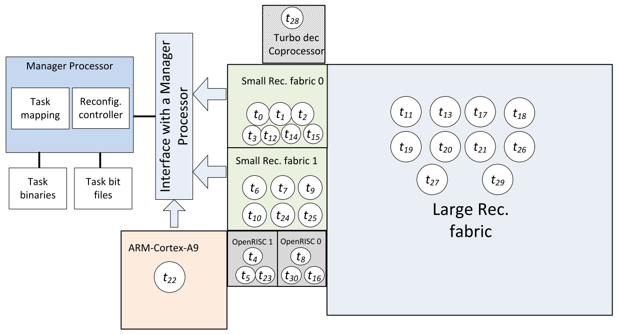


Fig. 2. Generated platform with the corresponding task mapping.

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

[1] M. Shafique, L. Bauer, and J. Henkel, "REMiS: Run-time energy minimization scheme in a reconfigurable processor with dynamic power-gated instruction set," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des. ICCAD Tech. Papers*, Nov. 2009, pp. 55–62.

[2] J. Castrillon, R. Leupers, and G. Ascheid, "MAPS: Mapping concurrent dataflow applications to heterogeneous MPSoCs," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 527–545, Feb. 2013.

[3] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "A methodology for mapping multiple use-cases onto networks on chips," in *Proc. Conf. Des., Autom. Test Eur.*, 2006, pp. 118–123.

[4] A. Schranzhofer, J.-J. Chen, and L. Thiele, "Dynamic power-aware mapping of applications onto heterogeneous MPSoC platforms," *IEEE Trans. Ind. Informat.*, vol. 6, no. 4, pp. 692–707, Nov. 2010.

[5] H. Yu, Y. Ha, and B. Veeravalli, "Quality-driven dynamic scheduling for real-time adaptive applications on multiprocessor systems," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2026–2040, Oct. 2013.

[6] C. Marcon, E. Moreno, N. Calazans, and F. Moraes, "Comparison of network-on-chip mapping algorithms targeting low energy consumption," *IET Comput. Digital Tech.*, vol. 2, no. 6, pp. 471–482, Nov. 2008.

[7] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 24, no. 4, pp. 551–562, Apr. 2005.

[8] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proc. Des., Autom. Test Eur. Conf. Exhibit.*, vol. 2. Feb. 2004, pp. 896–901.

[9] K. Srinivasan, K. Chatha, and G. Konjevod, "Linear-programming-based techniques for synthesis of network-on-chip architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 4, pp. 407–420, Apr. 2006.

[10] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, 2013.

[11] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proc. 50th Annu. Des. Autom. Conf.*, 2013, pp. 1:1–1:10.

[12] W. Jang and D. Pan, "A3MAP: Architecture-aware analytic mapping for networks-on-chip," in *Proc. 15th ASP-DAC*, 2010, pp. 523–528.

[13] O. He, S. Dong, W. Jang, J. Bian, and D. Z. Pan, "UNISM: Unified scheduling and mapping for general networks on chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1496–1509, Aug. 2012.

[14] D. Demirbas, I. Akturk, O. Ozturk, and U. Güdükbay, "Application-specific heterogeneous network-on-chip design," *Comput. J.*, vol. 4, no. 4, pp. 240–245, 2013.

[15] A. Hussien, A. Eltawil, R. Amin, and J. Martin, "Energy aware task mapping algorithm for heterogeneous MPSoC based architectures," in *Proc. IEEE 29th ICCD*, Oct. 2011, pp. 449–450.

[16] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[17] M. Vestias and H. Neto, "Co-synthesis of a configurable SoC platform based on a network on chip architecture," in *Proc. Conf. Asia South Pacific Des. Autom.*, Jan. 2006, pp. 1–6.

[18] H. Hedberg, T. Lenart, and H. Svensson, "A complete MP3 decoder on a chip," in *Proc. IEEE Int. Conf. MSE*, Jun. 2005, pp. 103–104.

[19] E. Grayver *et al.*, "Design and VLSI implementation for a WCDMA multipath searcher," *IEEE Trans. Veh. Technol.*, vol. 54, no. 3, pp. 889–902, May 2005.

[20] H. Harada, "Software defined radio prototype for W-CDMA and IEEE802.11a wireless LAN," in *Proc. IEEE 60th Veh. Technol. Conf.*, vol. 6. Sep. 2004, pp. 3919–3924.

[21] R. Asghar and D. Liu, "Dual standard re-configurable hardware interleaver for turbo decoding," in *Proc. 3rd ISWPC*, May 2008, pp. 768–772.

[22] C.-C. Lee, C.-F. Liao, C.-M. Chen, and Y.-H. Huang, "Design of 4 × 4 MIMO-OFDMA receiver with precode codebook search for 3GPP-LTE," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2010, pp. 3957–3960.

[23] J. Huang, C. Buckl, A. Raabe, and A. Knoll, "Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems," in *Proc. 19th Eur. Int. Conf. Parallel, Distrib. Netw. Based Process. (PDP)*, Feb. 2011, pp. 447–454.