Exploration of Power Domain Partitioning with Concurrent Task Mapping and Scheduling for Application-Specific Multi-core SoCs

Bo Wang¹(✉), Aneek Imtiaz², Joachim Falk¹, Michael Gläß³, and Jürgen Teich¹

¹ Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany
bo.wang1102@gmail.com, {joachim.falk,juergen.teich}@fau.de

² Technische Universität München, Munich, Germany
aneekimtiaz@gmail.com

³ Universität Ulm, Ulm, Germany
michael.glass@uni-ulm.de

Abstract. This paper proposes a novel approach to explore the design space of Power Domain (PD) partitioning in the architecture definition phase of heterogeneous SoCs. By formulating an Integer Linear Program (ILP), task mapping and scheduling is determined concurrently while considering power-off dependencies among cores in the same PD and the power-gating break-even time. Compared to state-of-the-art approaches aiming at design phases where task mapping and scheduling has been frozen, our proposed approach shifts joint exploration into earlier design phases, creates more power-gating opportunities for PD partitions, and thus identifies better trade-offs in terms of energy consumption and design costs.

Keywords: Power domain partition · Task mapping and scheduling · Evolutionary algorithm · Integer linear programming

1 Introduction

Power gating is an effective technique to reduce static power consumption of System-on-Chips (SoCs), like 5G New Radio modems in which dozens of heterogeneous cores are often adopted to achieve Gbits/s uplink and downlink speed. An SoC is divided into multiple Power Domains (PDs), which can be switched off individually when all cores and Hardware (HW) IPs in the same PD are idle, a so-called common idle interval. Power-gating control is more flexible when finer-grained power domains are partitioned. However, this would indeed result in a huge design, verification, and layout effort, even increase area and degrade power consumption and timing closure [13]. On the other hand, due to

This research work was funded by Intel Deutschland GmbH.

© Springer Nature Switzerland AG 2020

A. Brinkmann et al. (Eds.): ARCS 2020, LNCS 12155, pp. 153–167, 2020.

https://doi.org/10.1007/978-3-030-52794-5_12

Fig. 1. Task graphs for given applications V_0 and V_1 with the same period T_P but different deadline T_D , as well as underlying heterogeneous architecture R .

parallelism among tasks, merging HW resources which are active simultaneously into the same power domain may reduce design complexity without sacrificing power efficiency.

Some researchers have started investigating methodologies for exploration of PD partitioning to trade off energy consumption and the number of PDs. In [13], PD partitioning is explored by using a Multi-Objective Evolutionary Algorithm (MOEA), but it aims at the design phases in which task mapping and scheduling has been accomplished already, and determines the idle intervals of each HW resource rather than optimizing them. During subsequent PD partitioning, common idle intervals are post-processed for each PD, as well as power-gating break-even times. Power gating is exploited only for common idle intervals longer than a break-even time. After that, energy consumption is evaluated for partition candidates. Finally, trade-off fronts are obtained by the MOEA in terms of energy consumption and the number of used power domains. However, this approach does not explore the influence of task mapping and scheduling. We illustrate the lost potential through a motivating example in the following.

1.1 Motivating Example

Figure 1 shows two periodic applications with the same period generated by TGFF [5], as well as a HW architecture consisting of three fully connected heterogeneous processors. Power consumption of each processor is modeled by three power states [14], i.e., P_{run} , P_{idle} , and P_{off} , where the state *RUN* denotes the resource actively executing a program task, *IDLE* denoting being powered on with no task in execution, and *OFF* denoting the power-gated mode. First, task mapping and scheduling is performed to minimize energy consumption, where only two states – *RUN* and *IDLE* – and transition energy between them are assumed for processors. After that, PD partitions are explored using the approach in [13]. The found trade-off fronts are presented in Fig. 2. Take the trade-off front with 2 PDs as an example shown in Fig. 3(a). Although r_b is idle from 0 ms to 23 ms, PD_1 cannot be powered off because r_a is still executing. If the task mapping and scheduling would consider the power-off dependency between r_a and r_b , it may re-allocate the tasks and align the execution in the same PD.

Unfortunately, scheduling before PD partitioning does not have such knowledge and, thus, misses optimization potential.

Based on this observation, we propose a methodology to explore power domain partitioning with concurrent task mapping and scheduling. For each candidate explored during PD partitioning, task mapping and scheduling is performed with additional constraints for the power domain dependency and power-gating break-even time. As a result, more and longer common idle intervals in each PD may be created by properly mapping and aligning task execution on processors, as shown for v_{03} in Fig. 3. Power consumption is thus reduced due to longer power-gated state as shown in Fig. 3(b) and Fig. 2. More important, system architects may even prefer the 2-PD option identified by our approach to reduce design cost if it already meets the power target. The proposed approach actually expands the exploration space of PD partitioning.

Fig. 2. Power domain partitioning and task mapping & scheduling for the trade-off front with 2 PDs for the motivating example: (a) PD partitioning performed after task mapping and scheduling; (b) PD partitioning with concurrent task mapping and scheduling as proposed in this work.

Fig. 3. Trade-off fronts for normalized energy (to energy of 1-PD trade-off front obtained by partitioning PDs after task mapping and scheduling) vs. number of power domains (design complexity).

1.2 Contribution

State-of-the-art approaches for PD partitioning exploration consider power vs. design cost for heterogeneous multi-core SoCs where task mapping and scheduling has been already frozen at design time, e.g., assuming multiple subsystems are re-used and integrated in an SoC. This paper discusses the further optimizations applicable to SoCs when task mapping and scheduling can be combined with PD partitioning and jointly optimized. Our major contributions are summarized as follows:

- Tasks are mapped and scheduled specifically for each PD partition candidate, concurrently with PD partitioning exploration by a Multi-Objective Evolutionary Algorithm (MOEA). This aligns task execution and creates more common idle intervals for power gating.
- Task mapping and scheduling is formulated as an Integer Linear Programming (ILP), in particular integrating: 1) power-on/off dependencies introduced by PD partitioning among HW resources in the same PD; 2) constraints of power-gating break-even time due to transition energy and latency overhead.
- Experimental results show that our proposed joint exploration can identify much better trade-off fronts with significantly reduced design costs but without scarifying the power target. E.g., one experiment shows that the same power target can be achieved by 2 PDs, instead of 8 PDs when applying the approach in [13].

The aimed application domains of this work are time-critical or safety-critical [3] application specific embedded systems, such as wireless communications and electric vehicles [10]. There, most application tasks and use cases are known at design time, and static scheduling is also more favorable due to its determinism.

2 Related Work

Several research works exist on how to partition power domains at circuit level. In [2], Finite State Machine with Datapath (FSMD) circuits are decomposed into loosely coupled domains which may be power or clock gated. But, the workload characteristics are not considered. In [1], an approach leveraging rule-based design is proposed to automatically partition combinational logic into multiple PDs while considering usage characteristics. However, all of these studies [1,2] focus on micro-architecture level and RTL design phases. In [13], PD partitioning is explored at the Electronic System Level (ESL), thus for SoC architecture definition phases, but after task mapping and scheduling has been accomplished. As motivated earlier, this may hinder the maximization of idle intervals to reduce power or to allow to lower the number of power domains. In [7], a relevant task mapping and scheduling problem is discussed to Maximize Common Idle Interval (MCIT) among all cores, though the objective is to reduce active time and power consumption of a shared memory. The ILP formulation of common idle intervals is based on a discrete time axis. In [6], the idle interval of each core is modeled

at a continuous time axis. But the approach does not formulate common idle intervals. In both [7] and [6], homogeneous multi-core systems are considered. However, these are different from the power optimization of a heterogeneous architecture. Allocating tasks to more energy efficient cores may lead to lower power than merely pursuing MCIT. Moreover, both works do not investigate PD partitioning problem, but assume each core in an individual power domain.

Fig. 4. The proposed design flow for exploration of power domain partitioning with concurrent task mapping and scheduling.

Some other works address voltage-frequency islands partitioning at system level [9, 12], to reduce dynamic power. But the problem formulation is different. PD partitioning has to model power-off states, on/off dependencies within power domains, and power-off break-even times. This is difficult to model together with the problem of task mapping and scheduling. And, [12] does not consider task mapping and scheduling while [9] considers scheduling but not mapping.

3 Overview of the Methodology

An overview of our methodology is presented in Fig. 4.

Given

- Periodic applications, each of which can be modeled as a directed acyclic task graph $G(V, E, T_P, T_D)$, in which a task v belongs to the set of tasks V , E denotes data dependencies among tasks, an arbitrary period T_P and deadline T_D .
- An SoC architecture consisting of a set of HW resources denoted as R , power model of any resource $r \in R$ in different power states, e.g., $P_{run,r}$, $P_{off,r}$, $P_{idle,r}$, power-gating transition latency $T_{tr_off}(r)$ and energy $E_{tr_off,r}$, as well as wake-up transition latency $T_{tr_on,r}$ and energy $E_{tr_on,r}$ from power-off state.

- Mapping constraints that represent which task can be realized on which resource and the execution time of each task $D_{v,r}$ for a given resource.

Objective and Solution

The objective is to explore trade-off fronts in terms of energy consumption and the number of power domains (representing a measure of design complexity) for the problem of power domain partitioning including task mapping and scheduling.

An MOEA in [11] is used to explore the space of PD partitionings. Physical design or floorplan constraints can be added to prune the exploration space, if they can be forecast from previous products. For example, two resources far away in floorplan make less sense to be placed into the same PD.

For each PD partition, an ILP is generated and solved to determine a mapping and schedule for each task and a suitable schedule of power mode transitions for each PD, with the objective to minimize the energy consumption. Power-on/off dependencies of HW resources in the same PD, power-gating transition energy and break-even time are all considered here. The energy consumption value derived by the ILP solver is fed back to the MOEA as one evaluated objective of each PD partition. The state-based power modeling approach as in [14] is chosen because it achieves sufficient accuracy at system level and early design phases. The power models can be refined along the design cycle, e.g., consider different active power $P_{run,r}$ for different types of tasks running on a resource.

This work considers only static scheduling at design time. In principle, it may inspire the solution that considers the impact of run-time task migration. For example, add online scheduling algorithms after the ILP solver, and then evaluate the power consumption of each PD partition. However, it would take significantly longer exploration time, because the simulation is required to evaluate the power consumption. This is not the target application domain of this work.

4 ILP Formulation

The time is assumed to be discrete, divided into unit time intervals $[t, t + 1)$, for $t = 0, 1, \dots$, which we call *time slots* [7]. We refer to $[t, t + 1)$ as *time slot* t , or even as *time* t . Tasks are assigned to time slots and $D_{v,r}$ is an integer. The continuous-time version of the same problem can be approximated as a discrete-time version.

In this work, multiple independent periodic applications, e.g., $[V_0, \dots, V_L]$, with arbitrary deadlines and periods, can be considered together in a single ILP. This applies to the architecture which supports multiple applications simultaneously. A hyper-period of all applications, denoted as M , is chosen to map and schedule tasks from all applications within this hyper-period. Moreover, when the deadline of an application is longer than the period, a pipelined schedule is performed, i.e., a task graph is divided into several pipeline stages so the

current iteration of the task can overlap in execution with previous iterations [15]. However, our methodology is not limited to any specific pipeline approach, which is also not the focus of this paper. The following formulations are elaborated by using only one application with multiple periods for ease of explanation. But experiments in this work were done for problems containing multiple applications.

Table 1 defines ILP constants which are determined for each PD partitioning candidate by the MOEA. Table 2 explains the introduced ILP binary variables prior to introducing the ILP mapping and scheduling model.

Table 1. Constants in ILP formulation related to power gating and PD partitioning, and determined by the MOEA.

Symbols	Description
pd	A power domain from power domain set PD
$pd(r)$	The power domain containing resource r
$R(pd)$	Set of all resources in power domain pd
$T_{be,pd}$	Break-even time of power domain pd
$T_{tr_on,pd}$	Off-on transition time of power domain pd
$T_{tr_off,pd}$	On-off transition time of power domain pd
$E_{tr_on,pd}$	Off-on transition energy of power domain pd
$E_{tr_off,pd}$	On-off transition energy of power domain pd

Table 2. Binary variables in ILP formulation related to power gating and PD partitioning.

Symbols	Description
$X_{k,v,r}$	1 iff task v is mapped to resource r in period k
$S_{k,v,t}$	1 iff task v in period k is starting at time t
$B_{r,t}$	1 iff resource r is busy at time t
$C_{pd,t}$	1 iff all resources in power domain pd are mutually idle at time t
$I_{pd,t}$	1 iff all resources in power domain pd are mutually idle from time t to time $t + T_{be,pd} - 1$
$O_{pd,t}$	1 iff power domain pd is in off state at time t
$Z_{pd,t}$	1 iff power domain pd has either on-off or off-on transition at time t

4.1 Objective Function

An application with N periods is to be scheduled on a heterogeneous architecture. The interval of time slots is denoted as $T = [0 \dots M - 1]$, where the

hyper-period $M = NT_P$ in case of only one application. The objective function of the ILP is to minimize the total energy consumption according to Eqs. (1)–(8), including energy consumption of each resource in power states *RUN*, *IDLE* and *OFF*, denoted as $E_{run,r}$, $E_{idle,r}$ and $E_{off,r}$, as well as total on-off and off-on transition energies of each power domain, denoted as $E_{tot,tr,pd}$.

$$\text{minimize: } \sum_{r \in R} (E_{run,r} + E_{idle,r} + E_{off,r}) + \sum_{pd \in PD} E_{tot,tr,pd} \quad (1)$$

$$E_{run,r} = \sum_{t \in T} B_{r,t} * P_{run,r}, \quad \forall r \in R \quad (2)$$

$$E_{idle,r} = \left(\sum_{t \in T} (1 - B_{r,t}) - \sum_{t \in T} O_{pd(r),t} \right) * P_{idle,r}, \quad \forall r \in R \quad (3)$$

$$E_{off,r} = \left(\sum_{t \in T} O_{pd(r),t} - \frac{1}{2} * J_{pd(r)} \right) * (T_{tr_off,pd(r)} + T_{tr_on,pd(r)}) * P_{off,r}, \quad \forall r \in R \quad (4)$$

Transition energy of a power domain is calculated by Eqs. (5)–(8). J_{pd} denotes total number of transitions (both on-off and off-on) in power domain pd . The PD transition latency is determined by the resource with the longest latency in this PD. During power-off and power-on transitions, other resources are assumed to be in *OFF* state and *IDLE* state after its own transition, respectively. The related energies are modeled as part of the PD transition energy, as calculated by

$$E_{tr_off,pd} = \sum_{r \in R(pd)} \left(E_{tr_off,r} + (T_{tr_off,pd} - T_{tr_off,r}) * P_{off,r} \right), \forall pd \in PD \quad (5)$$

$$E_{tr_on,pd} = \sum_{r \in R(pd)} \left(E_{tr_on,r} + (T_{tr_on,pd} - T_{tr_on,r}) * P_{idle,r} \right), \forall pd \in PD \quad (6)$$

$$E_{tr,pd} = E_{tr_off,pd} + E_{tr_on,pd}, \quad \forall pd \in PD \quad (7)$$

$$E_{tot,tr,pd} = \frac{1}{2} * J_{pd} * E_{tr,pd}, \quad \forall pd \in PD \quad (8)$$

4.2 Constraints

Here, we focus on explanation of ILP formulation related to power gating and PD partitioning. Other ILP constraints for basic task mapping and scheduling are not elaborated, since they are very well-known and not novel, e.g. task mapping constraints, task dependency constraints, deadline constraints, and so on [8].

Unique Start Time Constraint: Each task must start exactly once, thus in one time slot.

$$\sum_{t \in T} S_{k,v,t} = 1, \quad \forall k \in [1 \dots N], \forall v \in V \quad (9)$$

Resource Busy Time Constraint: The number of busy slots of a resource should be equal to the total execution time of all tasks mapped on it. Moreover, from the start time slot of a task, it should be consecutive 1's assigned to the busy vector of a resource on which the task is mapped.

$$\sum_{t \in T} B_{r,t} = \sum_{k=1}^N \sum_{v \in V} X_{k,v,r} * D_{v,r}, \quad \forall r \in R \quad (10)$$

$$\begin{aligned} \sum_{i=t}^{t+D_{v,r}-1} B_{r,i} &\geq D_{v,r} * (S_{k,v,t} + X_{k,v,r} - 1), \\ \forall v \in V, \forall r \in R, \forall k \in [1 \dots N], \forall t \in [0 \dots M - D_{v,r}] \end{aligned} \quad (11)$$

Common Idle Time Constraint: A power domain is idle only when all resources in that domain are idle. This can be modeled by performing a logical NOR operation among the busy vectors of all resources in that domain:

$$C_{pd,t} = \neg \left(\bigvee_{r \in R(pd)} B_{r,t} \right), \quad \forall t \in T, \forall pd \in PD \quad (12)$$

The NOR operation is nonlinear, but the Boolean logic operation can be transformed to linear constraints. Let $N_R(pd)$ denote the number of resources in power domain pd . Equation (12) is transformed as below.

$$\sum_{r \in R(pd)} (1 - B_{r,t}) - N_R(pd) * C_{pd,t} \geq 0, \quad \forall t \in T, \forall pd \in PD \quad (13)$$

$$\sum_{r \in R(pd)} (1 - B_{r,t}) - N_R(pd) * C_{pd,t} \leq N_R(pd) - 1, \quad \forall t \in T, \forall pd \in PD \quad (14)$$

Off State Time Constraint: A power domain should be switched off only when its common idle interval is longer than its power-gating break-even time

which can be modeled as Eqs. (15)–(17), and rounded to the nearest greater integer.

$$T_{tr,pd} = T_{tr.off,pd} + T_{tr.on,pd} \quad (15)$$

$$T_{be.p,pd} = \frac{E_{tr,pd} - T_{tr,pd} \sum_{r \in R(pd)} P_{off,r}}{\sum_{r \in R(pd)} (P_{idle,r} - P_{off,r})} \quad (16)$$

$$T_{be,pd} = \lceil \max \{T_{tr,pd}, T_{be.p,pd}\} \rceil \quad (17)$$

To derive off-state slot vectors, an auxiliary variable $I_{pd,t} = 1$ is introduced to represent $T_{be,pd}$ adjacent slots of a pd from slot t to slot $t + T_{be,pd} - 1$ are all idle. This can be done by a logical AND operation:

$$I_{pd,t} = \bigwedge_{i=t}^{t+T_{be,pd}-1} C_{pd,i}, \quad \forall t \in [0 \dots M - T_{be,pd}], \forall pd \in PD \quad (18)$$

And, $T_{be,pd} - 1$ zeros have to be padded at the beginning and the end of vector $I_{pd,t}$ using Eq. (19).

$$I_{pd,t} = 0, \quad \forall pd \in PD, \forall t \in \{[1 - T_{be,pd} \dots - 1], [M - T_{be,pd} + 1 \dots M - 1]\} \quad (19)$$

Now, the final off state time slot vector $O_{pd,t}$ can be derived from Eq. (20). The off state slot $O_{pd,t} = 1$ if any of $I_{pd,t} = 1$ from slot $t - T_{be,pd} + 1$ to slot t . It can be performed by a logical OR operation. Equations (18) and (20) are non-linear, but they can be transformed into linear inequalities in a similar way as shown in Eq. (12). The details are not shown here.

$$O_{pd,t} = \bigvee_{i=t-T_{be,pd}+1}^t I_{pd,i}, \quad \forall t \in T, \forall pd \in PD \quad (20)$$

Transition State Time Constraint: On-off and off-on transition states are formulated by taking logical XOR operation of the current and previous one slot in the off state vector, as given in Eqs. (21). Similarly, it can be transformed into linear inequalities as well. The number of power domain transitions, i.e., J_{pd} , includes both off-on and on-off.

$$Z_{pd,t} = O_{pd,t-1} \oplus O_{pd,t}, \quad \forall t \in [1 \dots M - 1], \forall pd \in PD \quad (21)$$

$$J_{pd} = \sum_{t \in [1 \dots M - 1]} Z_{pd,t}, \quad \forall pd \in PD \quad (22)$$

5 Experimental Results

The proposed approach has been experimented on different benchmarks. The first set of benchmarks is from a public benchmark suite E3S [4], while the second one consists of synthetic benchmarks generated using the tool TGFF [5]. The main program of the flow was implemented using Python, but the MOEA was implemented using Java [11]. All of programs have been executed on a laptop with an i5-5300U CPU @ 2.3 GHz (2 cores, 4 threads) and 12 GB DDR memory.

For comparison, the same experiments were performed by applying the approach [13] performing PD partitioning after task mapping and scheduling. We called it as the reference approach in the following. Here, various task mapping and scheduling algorithms can be applied before PD partitioning with desired optimization objectives, like execution time or power. They lead to different energy consumption after PD partitioning and power gating. Since our work focuses on energy optimization, as a fair comparison, we performed an energy-aware task mapping and scheduling also using the approach of ILP. But in this ILP formulation, processors are assumed to be only in *RUN* or *IDLE* states without *OFF* states. PD partitioning and power-gating related constraints are not applied during this step. Therefore, in the objective function, $E_{off,r}$ and $E_{tot,tr,pd}$ in Eq. (1) become zero, and $\sum_{t \in T} O_{pd(r),t}$ in Eq. (3) are zero too.

Fig. 5. Trade-off fronts for E3S benchmarks with normalized energy (to energy of 1-PD partition obtained by the reference approach [13], i.e., PD partitioning performed after mapping and scheduling) vs. hardware complexity (number of power domains).

5.1 Benchmark Applications from E3S

Three benchmarks are selected from E3S [4], i.e., Networking, Telecom and Consumer. They are scheduled onto a heterogeneous architecture consisting of a 2-D 3×3 mesh of processors whose power consumption is also specified in E3S. The

transition latency in Table 1 varied in the range of 10–50 us, and the task execution times ranged in the interval of 0.5–1 ms. The transition energies $E_{tr_off,r}$ and $E_{tr_on,r}$ in Eqs. (5)–(6) were assumed zero in the following. Therefore, the power-gating break-even time $T_{be,pd}$ was determined by the transition latency $T_{tr,pd}$ according to Eqs. (16)–(17).

The MOEA has been configured to use 20 generations with 10 individuals per generation. For each number of power domains, the solutions with the lowest normalized energy according to Eqs. (1)–(8) are shown in Fig. 5.

It can be noticed that for each number of power domains, the trade-off point using our approach has a lower energy. This is because our concurrent mapping and scheduling of tasks with PD partitioning is able to create more common idle intervals specific for each PD partition to allow more power-off opportunities. Therefore, better power savings can be achieved even with fewer power domains. For example, in the benchmark of Telecom, a lower power consumption can be achieved even with 2-PD partition, in comparison to a trade-off point for an 8-PD partition in the reference approach [13]. Much better PD partitioning trade-off points can be identified to meet the power target at significantly reduced design cost.

The exploration took 8–9 h in which we set the timelimit of the ILP solver to 3 min. Notably, this is longer than the approach [13] which took about 1–2 h. This is expected, because our approach has to perform task mapping and scheduling in addition. Still, limiting the ILP solver to 3 min has two impacts: 1) the currently best found solution by the ILP may not be the optimal one in terms of energy consumption, but has a relative optimality gap of 10–20%, reported by the solver; 2) the ILP solver even may not find any feasible solution as the problem size increases, though it never happened in our benchmarks. Nevertheless, our approach was always able to find lower energy consumption points for each number of PD partitions. If more exploration time is acceptable, our approach would be able to probably find even better results. This is a trade-off that system architects can decide during system-level exploration.

Table 3. Three use cases generated by TGFF.

Use case	Application	Period	Deadline	Hyper-period	Iterations
1	V_0	9 ms	9 ms	18 ms	2
	V_1	18 ms	18 ms		1
2	V_0	7.5 ms	6 ms	15 ms	2
	V_1	15 ms	12 ms		1
	V_2	7.5 ms	6 ms		2
3	V_0	5 ms	10 ms	10 ms	2
	V_1	10 ms	15 ms		1
	V_2	10 ms	12 ms		1

5.2 Benchmarks Generated by TGFF

Three benchmarks have been generated by TGFF [5], with different tightness of deadline, i.e., the deadline is equal to, shorter, or longer than the period, as shown in Table 3. Each use case has multiple applications to be scheduled over their hyper-period. When the deadline is longer than the period, e.g., in use case 3, or a multimedia streaming application, different iterations of applications can overlap. Therefore, we partitioned the task set and performed scheduling for steady state in one hyper-period [15]. All three use cases have size in the range of 40–50 tasks whereas the heterogeneous architecture consists of a 2D mesh with 3×3 processors. The power data, transition time, and energy for these processors were obtained from an in-house design. The EA parameters for the exploration are the same as for the E3S benchmarks. As shown in Fig. 6, our proposed approach also identifies better trade-off fronts than the reference approach [13].

Fig. 6. Trade-off fronts for TGFF benchmarks with normalized energy (to energy of 1-PD partition obtained by the reference approach [13], i.e., PD partitioning performed after mapping and scheduling) vs. hardware complexity (number of power domains).

5.3 Scalability Analysis

The total exploration time depends on two parts: 1) EA parameters, mainly the number of generations and individuals per generation (PD partition options), which typically increases with the higher complexity of the hardware architecture; 2) execution time of ILP solver for each PD partition option, which scales non-linearly with the size of hardware architecture, the size of task graph, and most importantly, with the time scale of the schedule. Therefore, our approach is not easily scalable for bigger problems. We experimented the execution time of the ILP solver, given an architecture of a mesh network with 6 processors. When increasing the number of tasks to 80 and set the relative optimality gap of

the ILP solver to 20%, a feasible schedule cannot be found within 2 h though it is preferred in the range of minutes as a part of whole flow. Alternative models for scheduling might be the key to reduce the number of binary variables and thus search space of the ILP formulation to improve scalability.

Although our performed experiments were solvable for real-world benchmarks in still an acceptable amount of time, we envision to investigate scalability in future work.

6 Conclusion

In this paper, an exploration approach is proposed to systematically explore PD partitioning for heterogeneous multi-core SoCs, jointly with task mapping and scheduling. An ILP-based task mapping and scheduling is performed for each PD partition candidates while partitioning PDs by a Multi-Objective Evolutionary Algorithm. The ILP formulation considers the constraints of power-off dependencies among hardware resources belonging to the same PD and the power-gating break-even time. For a given PD partition, it creates more and longer common idle intervals of PDs which can be switched off more often to save power. Compared to state-of-the-art approaches performed after task mapping and scheduling frozen, our approach offers significantly larger optimization opportunities for system architects. It has been shown that better trade-off fronts in terms of energy consumption and number of PDs and thus hardware costs may be found by shifting exploration to earlier design phases.

Acknowledgments. This research work was funded by Intel Deutschland GmbH, and finished before Bo Wang and Aneek Imtiaz left Intel. We would like to acknowledge Dr. Yang Xu and Dr. Ralph Hasholzner at Intel, and also Dr. Thomas Wild and Prof. Andreas Herkersdorf at Technische Universität München, for valuable discussions.

References

1. Agarwal, A., Arvind, A.: Leveraging rule-based designs for automatic power domain partitioning. In: ICCAD, November 2013
2. Agarwal, N., et al.: FSMD partitioning for low power using simulated annealing. In: ISCAS, May 2008
3. Baruah, S., Fohler, G.: Certification-cognizant time-triggered scheduling of mixed-criticality systems. In: RTSS, November 2011
4. Dick, R.: Embedded system synthesis benchmarks suites (E3S) (2017). <http://ziyang.eecs.umich.edu/~dickrp/e3s/>
5. Dick, R., Rhodes, D., Wolf, W.: TGFF: task graphs for free. In: CODES/CASHE, March 1998
6. Esmaili, A., Nazemi, M., Pedram, M.: Modeling processor idle times in MPSoC platforms to enable integrated DPM, DVFS, and task scheduling subject to a hard deadline. In: ASPDAC, January 2019
7. Fu, C., Zhao, Y., Li, M., Xue, C.J.: Maximizing common idle time on multicore processors with shared memory. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2017)

8. Glaß, M., Teich, J., Lukasiwycz, M., Reimann, F.: Hybrid optimization techniques for system-level design space exploration. In: Ha, S., Teich, J. (eds.) *Handbook of Hardware/Software Codesign*. Springer, Dordrecht (2017). https://doi.org/10.1007/978-94-017-7267-9_8
9. Liu, Y., Yang, Y., Hu, J.: Clustering-based simultaneous task and voltage scheduling for NoC systems. In: ICCAD, November 2010
10. Lukasiwycz, M., et al.: Cyber-physical systems design for electric vehicles. In: DSD, September 2012
11. Lukasiwycz, M., Glaß, M., Reimann, F., Teich, J.: Opt4J-a modular framework for meta-heuristic optimization. In: GECCO, July 2011
12. Ogras, U.Y., Marculescu, R., Choudhary, P., Marculescu, D.: Voltage-frequency island partitioning for GALS-based networks-on-chip. In: DAC, June 2007
13. Wang, B., et al.: Exploration of power domain partitioning for application-specific SoCs in system-level design. In: GI/ITG/GMM Workshop Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, MBMV, March 2016
14. Xu, Y., et al.: A very fast and quasi-accurate power-state-based system-level power modeling methodology. In: Herkersdorf, A., Römer, K., Brinkschulte, U. (eds.) ARCS 2012. LNCS, vol. 7179, pp. 37–49. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28293-5_4
15. Yang, H., Ha, S.: Pipelined data parallel task mapping/scheduling technique for MPSoC. In: DATE, April 2009