Thermal-Aware Mapping of Streaming Applications on 3D Multi-Processor Systems

Marco Cox*, Amit Kumar Singh†, Akash Kumar† and Henk Corporaal*

*Department of Electrical Engineering
Eindhoven University of Technology, The Netherlands
Email: m.g.h.cox@student.tue.nl, h.corporaal@tue.nl

†Department of Electrical & Computer Engineering
National University of Singapore, Singapore
Email: {amit.singh, akash}@nus.edu.sg

***Abstract*—Implementing Multi-Processor-Systems-on-Chip (MPSoCs) in 3-Dimensional (3D) ICs has many benefits, but the increased power density can cause significant thermal problems, resulting in decreased reliability, lifetime and performance. This paper presents a fast thermal-aware approach for mapping throughput constrained streaming applications on a 3D MPSoC. While there are some published works on thermal-aware mapping of real-time applications, throughput constraints and data dependencies are mostly not considered. Further, conventional approaches have long running times due to slow iterative thermal simulations. In our approach, to avoid slow thermal simulations for every candidate mapping, a thermal model of the 3D IC is used to derive an on-chip power distribution that minimizes the temperature before the actual mapping is done. Next, this distribution is used in a resource allocation algorithm to derive a mapping that meets the throughput constraint while approaching the target power distribution and minimizing energy consumption. This way, in contrast to most existing approaches, a mapping can be derived in the order of minutes. Experiments show a 7% reduction in peak temperature and a 47% reduction in communication energy compared to mappings based on load balancing.**

I. Introduction

Three-dimensional (3D) integrated circuits provide interesting possibilities for implementing Multi-Processor-Systems-on-Chip (MPSoCs). In a 3D IC, multiple layers of logic are stacked vertically, and Through Silicon Vias (TSVs) are generally used for connecting the layers, as illustrated in Figure 1. Stacking multiple layers of processing and/or memory elements into a 3D IC can significantly reduce the die size and average interconnect wire length. This in turn can reduce production costs, communication delays and interconnect energy, while increasing the interconnect flexibility [1], [2]. However, stacking active layers increases the power density, which can cause serious thermal problems, affecting both the performance and reliability of the system. Higher temperatures increase power consumption due to leakage and decrease the system reliability and lifetime, even below the thermal emergency threshold. Thermal problems are regarded as one of the main limiting factors for future high performance systems [3], indicating the importance of minimizing them throughout the design process. When the temperature rises above the acceptable limit, protection mechanisms based on Dynamic Voltage/Frequency Scaling (DVFS), clock gating or task migration are required to protect the hardware [4]. These dynamic mechanisms can however have an unpredictable impact on the software execution, causing for example, missed deadlines. Therefore, especially for time-constrained applications, it is important to consider temperature when mapping applications in order to avoid unpredictable emergencies.

In this work, an integrated thermal-aware approach for mapping streaming applications on a 3D MPSoC is introduced. A streaming application, for example a video decoder, typically has a throughput requirement, which possibly cannot be guaranteed in combination with unexpected thermal emergencies in a (3D) IC. The goal of the proposed approach is to map and schedule multiple applications at design time, satisfying all throughput, communication and storage constraints, while minimizing the peak-temperature and temperature gradients across the chip. Since simulating the thermal process with a high spatial and temporal resolution is very time consuming, the approach described in this work is split into two main steps. In the first step, thermal characteristics of the 3D IC are extracted from a simple but flexible model of the physical chip. Then, the extracted profile is passed to the actual resource allocation algorithm, which doesn't require iterative temperature simulations. This causes the running time of the total flow to be in the order of minutes, in contrast to most existing thermal-aware mapping approaches.

The MPSoC is assumed to contain a 3D mesh of (homogeneous or heterogeneous) processing tiles. Since Networks-on-Chip (NoCs) are widely regarded as the most promising communication architecture for many-core 3D architectures [2], a 3D NoC is assumed to provide the communication between the tiles. The streaming applications are modeled as Synchronous Dataflow Graphs (SDFGs) [5]. Since there are multiple trade-offs involved, for example between optimizing for energy consumption or peak temperature in the 3D IC [6], a set of cost function weights is used to steer the optimization process by weighting the different optimization criteria. This results in a flexible thermal-aware mapping flow. The thermal

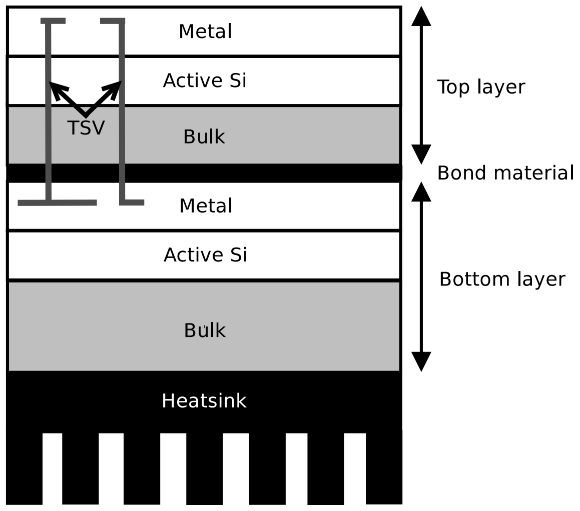

Fig. 1. Two stacked layers connected by TSVs

process resulting from the mapping is simulated using a modified version of the *HotSpot* thermal simulator [7], which also takes the thermal effect of TSVs into account.

The main contributions presented in this work are:

1) A fast algorithm is proposed to find a static power distribution within a 3D IC that minimizes the peak temperature. The effect of TSVs on the temperature distribution is taken into account.
2) A thermal-aware approach for mapping streaming applications on a 3D MPSoC is developed, taking both throughput constraints and data-dependencies into account.

To evaluate performance, the proposed flow is used to map and schedule a set of synthetic benchmark applications as well as real-life multimedia streaming applications on a NoC-based 3D MPSoC. The performance evaluations show that compared to the load balancing strategy, the peak temperature and communication energy are reduced by 7% and 47% respectively, while meeting all timing and storage constraints.

To the best of our knowledge, this is the first thermal-aware approach for mapping throughput-constrained applications on 3D MPSoCs.

Overview: In section II, related work regarding the mapping of (streaming) applications onto 2D and 3D MPSoCs is discussed. The architecture, application and thermal models are introduced in section III. The proposed thermal-aware mapping flow is discussed in section IV. Experimental results are presented in section V. Section VI concludes the paper.

II. Related Work

Thermal-aware mapping and scheduling on 3D multiprocessor systems is a well-studied topic. Multiple approaches have been proposed, which can be split into dynamic (run-time) and static (design-time) thermal-aware mapping techniques. Dynamic approaches generally try to measure or estimate the current temperature distribution in the chip, and take actions based on that in order to minimize hotspots and thermal gradients (spatial, temporal or both). In [4], several dynamic mechanisms such as temperature-triggered Dynamic Voltage/Frequency Scaling (DVFS), clock gating and hot task migration are reviewed, and a run-time task assignment algorithm is proposed that takes the thermal history of cores into account. A thermal-aware OS-level scheduler for 3D MPSoCs is proposed in [8]. These methods share the goal of minimizing the peak temperature and thermal gradients without sacrificing performance too much. Constraints such as deadlines or memory requirements are however not taken into account, as well as the effect of inter-task communication. Interconnect utilization should be taken into account since a NoC can dissipate a substantial part of the power budget, and the dissipation depends on the traffic [9]. For 2D ICs, a thermal-aware MPSoC assignment and scheduling technique for real-time applications is proposed in [10], which is based on mixed integer linear programming. However, the techniques for 2D ICs cannot simply be applied to 3D ICs due to the significantly different thermal behaviour of 3D ICs. Skadron et al. [7] developed the HotSpot thermal simulator to evaluate the steady state and dynamic temperature distribution in ICs.

Static mapping approaches aim at finding a thermal-aware mapping at design time, by using a model of the physical chip, or by using general knowledge about the thermal behaviour of 3D ICs. In [9], both temperature and communication load are considered, and a genetic algorithm is used to generate static mappings. Cheng et al. [6] show that a trade-off exists when minimizing power consumption as well as peak temperature, and use a combination of heuristics, simulated annealing and a greedy algorithm to find optimal static mappings. However, application constraints such as throughput requirements are not taken into account. The authors of [11] propose a technique to find optimal mappings in a thermal sense for applications with deadlines. First, a power balancing algorithm is used to find an initial mapping. The initial mapping is then iteratively improved by simulating the temperature distribution and migrating tasks. Communication between tasks as well as memory constraints are not taken into account. Thiele et al. [12] argue that being able to guarantee a bound on the peak temperature in a MPSoC is important, since it removes the need for unpredictable run-time mechanisms. Towards this, the authors use formal methods to find optimal static mappings of SDFG-modeled applications on heterogeneous 2D MPSoCs, while guaranteeing performance and peak temperature. The communication overhead is taken into account, but the power dissipation of the NoC has not been considered. Since vertical communication in a 3D NoC can be considerably faster and more energy efficient than horizontal communication [6], a straightforward extension of their methods to 3D MPSoCs will not provide optimal mapping solutions. In contrast to the above mentioned strategies, our approach performs thermal-aware mapping of throughput-constraint applications on 3D MPSoCs while taking memory as well as communication constraints into account. Further, our approach considers the effect of TSVs on the temperature distribution and power dissipation, and minimizes the communication energy.

Next to thermal-aware mapping, thermal-aware floorplanning can also help to reduce thermal problems. Thermal-aware floorplanning techniques have been developed for 2D ICs [13] as well as for 3D ICs [14], [15]. In this work, the floorplan is assumed to be known and fixed, for example generated by a thermal-aware floorplanner.

III. Preliminaries

This section covers the application model, multiprocessor platform model and the 3D IC model.

A. Application Model

To model streaming applications with a throughput constraint, Synchronous Dataflow Graphs (SDFGs) [5] are used. In an SDFG, an application is modeled as a set of tasks, called *actors*, that communicate chunks of data with a predefined size, called *tokens*. An example SDFG that models an H.263 decoder is depicted in Figure 2. The nodes correspond to the actors and the edges represent data dependencies, referred to as connections, between the actors. The H.263 decoder is modeled with four actors (*vld, iq, idct* & *mc*) and four edges (d_1, d_2, d_3 & d_4). An actor has fixed input and output rates on every connection. The input rate corresponds to the number of tokens that the actor consumes from the incoming connection when executed (fired) once. Similarly, the output rate defines the number of tokens that are produced on the outbound connection during one execution of the actor. An initial number of tokens might be available on the connection. An actor is executed as soon as sufficient tokens are available at all its incoming connections, and enough buffer space is available to store the produced tokens. The size of a token may be different for every connection.

The input and output rates of the actors in an SDFG determine the relative frequencies with which the actors can execute, which can be represented by a unique repetition vector. In the application model, the worst-case execution times (in time-units) and memory requirements (in bits) of all actors on all possible processing elements are specified. For example, an actor performing encoding may have a worst-case execution time of 10000 time-units on an ARM7 or 2000 time-units on dedicated encoder hardware. Specifying requirements for all possible mappings enables the use of heterogeneous architectures. For all connections, the size of the tokens (in bits) is specified, as well as the memory required when mapping the connection to memory, or the bandwidth required when mapping the connection to interconnect. If actors are fired as soon as they are ready to fire (self-timed execution), the execution pattern of a consistent, strongly connected SDFG is always periodic after an initial start up period [16]. The time between two recurring states in the execution of an SDFG defines the throughput of the application. An application may have a throughput requirement, fixing the maximum time between two recurring states. The throughput of an SDFG may be calculated by simulating the execution until a recurrent state is found [16].

B. Multiprocessor Platform Model

In this work, a regular 3D mesh of tiles connected by a Network-on-Chip (NoC) is considered, as depicted in Figure 3.

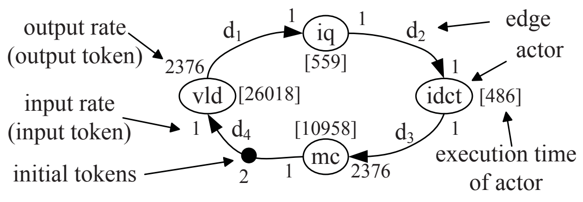

Fig. 2. SDFG of an H.263 decoder

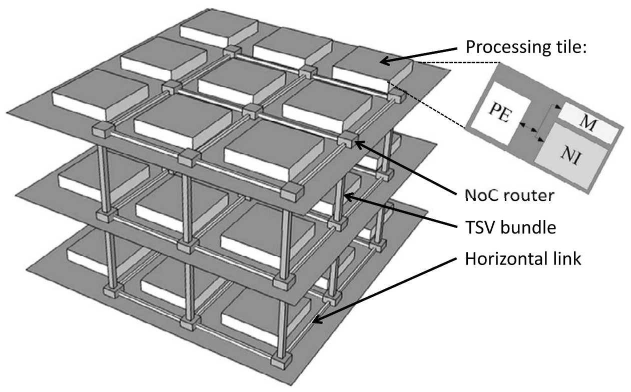

Fig. 3. Example 3D mesh of tiles.

Every tile contains at least a network interface (NI), connecting the tile to the interconnect network. Furthermore, a tile may contain a processing element (P) of some type PT (processor type), for example an ARM core, and a memory. Different types of processors are possible, allowing the modeling of heterogeneous architectures. Such an architecture can be modeled by an architecture graph consisting of tiles and connections as defined below.

Definition 1 (Tile): A tile t is a 9-tuple (pt, w, m, c, i, o, pa, pi, pm) with $pt \in PT$ the processor type, $w \in \mathbb{N}_0$ the TDMA time wheel size, $m \in \mathbb{N}_0$ the available memory (in bits), $c \in \mathbb{N}_0$ the maximum number of supported connections, $i, o \in \mathbb{N}_0$ the maximum i/o bandwidth (in bits), $pa, pi \in \mathbb{R}$ the active and idle power (in W) and $pm \in \mathbb{N}_0^3$ the position of the tile in the mesh.

Tiles are connected by a NoC. In the architecture graph, the NoC is abstracted to a set of point-to-point connections between tiles.

Definition 2 (Connection): A connection c is a 5-tuple (u, v, l, h_h, h_v) with $u \in T$ the source tile, $v \in T$ the destination tile, $l \in \mathbb{N}_0$ the latency (in time-units) and $h_h, h_v \in \mathbb{N}_0$ respectively the number of horizontal and vertical hops between u and v.

The set of tiles T and the set of connections C together define the architecture graph. A tile is assumed to consume pa W of power when active, and pi W when idle.

Vertical links between tiles are generally implemented using TSVs. As reported in [6], TSVs can often provide faster and more energy efficient communication compared to horizontal links, mainly because of their short length. The differences in delay and energy per bit depend on the technology and the NoC topology. For example, the NoC switches can simply be extended with 2 extra ports for up/down communication, or the vertical links can be implemented as a shared bus. This work does not treat all these options in detail, but the latency (l) and horizontal and vertical hop count (h_h, h_v) properties of a connection do provide some room for modeling different 3D NoC implementations in the architecture graph. For example, the latency of a connection can be defined such that it depends on the direction of the communication.

C. 3D IC model

To be able to simulate the on-chip temperature for a given execution trace, a model describing the thermal characteristics of the 3D IC is required. This model is also used to extract information about the thermal behaviour of the chip, which can be used by the mapping algorithm. The model used in this work is based on the 3D grid model available in the HotSpot thermal simulator [7]. The model contains all relevant physical properties of the IC and the heat sink, as well as a set of active and inactive layer specifications. Active layers correspond to layers that actually dissipate power, while inactive layers are used to model the bonds (glue, thermal interface material) between the active layers. Figure 4 illustrates the floorplan of an active layer. For every layer, the thickness, material properties and a floorplan are specified. Floorplans of the active layers consist of tiles, all of which contain one or more blocks (e.g. processor, memory, router), specified by the floorplan of the specific tile type. Execution of the application models is tracked at the tile level. The power that is dissipated in a tile, is assumed to be distributed over the blocks of that tile. For example, 80% may be dissipated in the processor block, 10% in the router block and 10% in the memory. Blocks can correspond to function blocks of the processor, but the processor can also be modeled as one block. This enables the use of both fine (detailed) and course grained models.

TSVs are generally made of copper, which has significantly different thermal properties than silicon. To take the thermal effect of TSVs into account, the size, position and material properties of the TSVs are also specified in the 3D IC model. To be able to simulate the thermal impact of the TSVs, the HotSpot simulator is extended to take TSVs into account. This is done by changing the thermal properties (conductance and heat capacity) of grid cells in the internal HotSpot model that contain TSV material, based on the ratio of the grid cell volume that is occupied by TSV material.

IV. Thermal-aware mapping

This section introduces the proposed mapping flow. The general structure of the flow is depicted in Figure 5. The flow consists of two main steps: a "*thermal profiling*" step and the actual mapping algorithm. In the "*thermal profiling*" step, the physical model of the 3D IC is used to derive a power distribution among the tiles that minimizes the peak temperature and spatial temperature gradients. For example, in the power distribution that minimizes the peak temperature, tiles that are on the layer closest to the heat sink are likely to dissipate more power than tiles far away from the heat sink. This is because the layers close to heat sink are able to get rid of the heat faster, therefore they are able to handle more

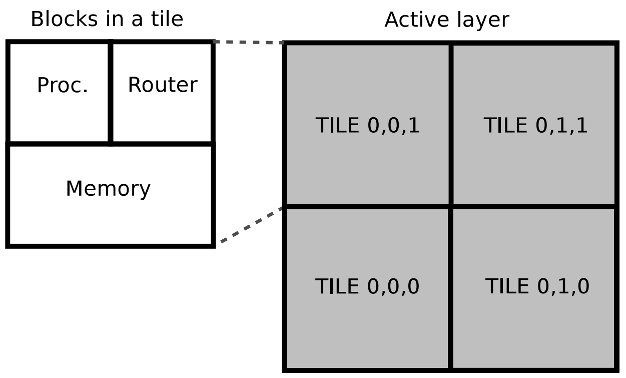

Fig. 4. Active layer floorplan illustration.

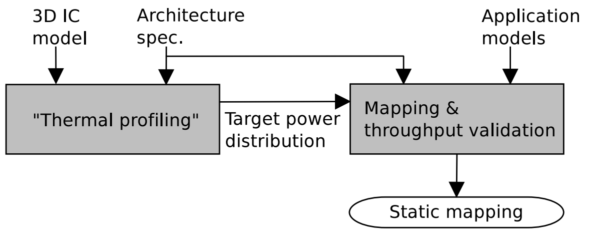

Fig. 5. Overview of the thermal-aware mapping flow.

load/power without overheating. Thus, they get a higher target power ratio. The power density, the floorplan and the absolute position of a tile in the horizontal plane will also influence its target power ratio. The resulting "*target power distribution*" assigns a power ratio to every tile, and is passed to the mapping algorithm. The mapping algorithm tries to find a mapping that approaches this power distribution while minimizing the energy and meeting all timing and storage constraints.

The two step structure is based on the observation that (high resolution) thermal simulations have a long running time, making it impossible to simulate the temperature for every candidate mapping within a limited running time. A lot of existing approaches avoid iterative thermal simulations by just applying heuristics to optimize for temperature. However, these heuristics are often not very accurate in a quantitative sense, requiring the designer to tune the heuristics by hand in order to match the actual chip properties and find good mappings. In our approach, this tuning is done automatically in the first step of the flow. In the mapping step, no thermal simulations are required, which drastically reduces the running time compared to methods that simulate the temperature for a lot of candidate mappings.

The remainder of this section discusses the steps of the flow in more detail.

A. Thermal profiling

The structure of the *thermal profiling* step is depicted in Figure 6. The update algorithm adjusts the power ratios R_t of the tiles based on the steady state temperature distribution resulting from the previous power distribution. The power ratio R_t of a tile corresponds to the ratio between the total chip power P and the power dissipated in tile t. The power

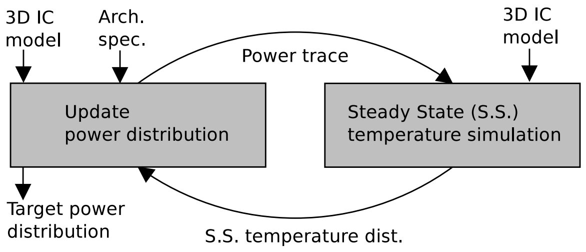
3D IC model
Arch. spec.
Power trace
3D IC model
Update power distribution
Steady State (S.S.) temperature simulation
Target power distribution
S.S. temperature dist.

Fig. 6. Structure of the "thermal profiling" algorithm.

dissipated in tile t, $P_t = R_t * P$, is distributed among the blocks in that tile based on an intra-tile power distribution, which may be constant in simple models. This way, a power trace is generated for every block in the chip. To limit the number of thermal simulations, a heuristic is used for updating the power distribution. The power ratios of tiles with a peak temperature above the average are decreased, while the power ratios of tiles with a peak temperature below the average are increased in the update step. This way, temperature differences among tiles are decreased, resulting in a lower peak temperature and smaller temperature gradients. The adaptation rate of the power ratios is defined by constant α. The update algorithm is summarized in Algorithm 1.

In agreement with observations described in related literature [4] [8], some general observations can be made regarding the power distribution after convergence of the algorithm:

1) Tiles on layers farther away from the heat sink get hotter than tiles closer to the heat sink with the same power dissipation. Hence, the algorithm will in general assign smaller power ratios to tiles farther away from the heat sink.
2) The thermal conductance in the vertical direction is generally high compared to the horizontal direction, because the layers are thin (typically 20-100 μm). As a result, blocks with a high power density that are stacked on top of each other generate high temperatures. Because of this, the combined power dissipation of horizontally aligned blocks in different layers will be limited.
3) TSVs increase the thermal conductance between layers. As a result, temperature differences between layers get decreased. The magnitude of this effect depends on the TSV material, size and density.
4) Blocks that are near the edges/corners of the die tend to get hotter than blocks farther away from the edges, since there is less material for the heat to spread to.

The above observations are used to develop heuristics in some thermal-aware mapping approaches, for example by assigning higher costs to mappings that use tiles that are further away from the heat sink, or by balancing computational load over "stacks" of vertically adjacent tiles [8]. However, these heuristics are generally not tuned to the specific IC that is considered, possibly resulting in suboptimal solutions. In our approach, the temperature-related observations are modeled implicitly in the *target power distribution*, which is derived directly from a model of the 3D IC. This results in assumptions that are more representative for the specific 3D IC that is considered.

ALGORITHM 1: Thermal Profiling Algorithm

```
Input: 3D IC model, stopping criterion $\delta \in \mathbb{R}$, max. # of iterations
       $I_{max} \in \mathbb{N}$, adaptation constant $\alpha \in \mathbb{R}$, total chip power
       $P \in \mathbb{R}$.
Output: Target power ratios $R_t$, $t \in [0, N_{tiles} - 1]$.
Initialize power ratios $\forall t \in [0, N_{tiles} - 1] : R_t = 1/N_{tiles}$;
$T_{avg} = 0$, $T_{prev.max} = \infty, T_{max} = 1000$;
$i = 0$;
while $(T_{prev.max} - T_{max}) \geq \delta$ and $i \leq I_{max}$ do
    Generate power traces for all blocks based on $P$, $R_t$ and the
    intra-tile power distribution;
    Simulate steady state temperature dist.;
    $T_{avg} \leftarrow$ average chip temperature;
    $\forall t \in [0, N_{tiles} - 1] : T_{peak,t} \leftarrow$ max. temp. in tile $t$;
    $T_{max} \leftarrow \max(T_{peak})$ ;
    if $(T_{prev.max} - T_{max}) < \delta$ then
        break
    end
    if $T_{prev.max} < T_{max}$ then
        $\alpha = \alpha/2$;
        Restart algorithm;
    end
    for all tiles $t \in [0, N_{tiles} - 1]$ do
        $d = (T_{peak,t} - T_{avg})/T_{avg}$;
        $R_t = \max(0, R_t * (1.0 - (\alpha * d)))$;
    end
    Renormalize power ratios $R_t$;
    $T_{prev.max} = T_{max}$;
    $i + +$;
end
return power ratios $R_t$
```

Running time: The steady state temperature is iteratively simulated by the modified HotSpot thermal simulator, which determines the running time of the algorithm. The simulation time depends on the spatial resolution and the number of layers. For an IC with 3 active layers and a grid resolution of 32×32, one simulation takes 117s on a 2.3GHz Intel i7 CPU (single threaded). With a well chosen value for the adaptation constant α, the algorithm converges to a static power distribution in 6-10 iterations, resulting in a total running time of up to 1170s. Note that the running time is independent from the number of tiles on a layer, since the thermal simulator internally uses a grid with a fixed resolution.

B. Application Mapping Flow

An overview of the mapping flow is depicted in Figure 7. The application graphs, the architecture specification and the target power distribution serve as inputs to the flow. For the memory dimensioning, constraint refinement and communication scheduling steps, existing implementations available in the SDF^3 tool set [17] are applied. The other steps are described subsequently.

Application merging: In practical situations, use cases consisting of multiple applications running simultaneously are common. To support the mapping of multiple applications, in the first step of the flow, all application graphs are merged into one application graph using the rate control principle. In

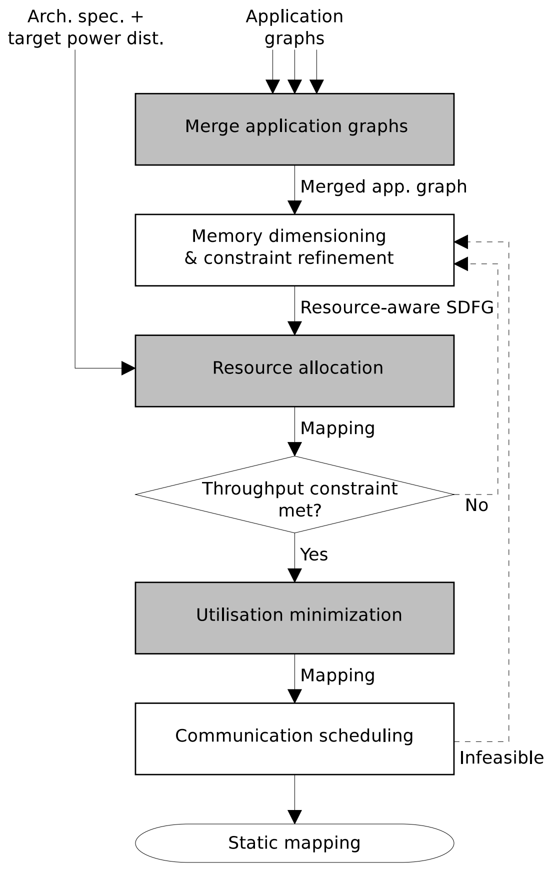

Fig. 7. Overview of the mapping flow.

this approach, a *rateControl* actor is inserted to control the relative execution rates of the different applications that are merged into one graph. Connections between the *rateControl* actor and one actor in every application are added, with input and output rates that force the execution rates of the applications to synchronize in a desired ratio. For example, application *A* might have a throughput requirement twice as high as that of application *B*. In that case, the rate controller will force the execution of application *B* to stall until *A* has executed twice, and vice versa. The throughput constraint of the merged application is the minimum of the original individual throughput constraints, and the individual applications can have throughput constraints that are a multiple of the overall throughput constraint.

Resource Allocation: In the resource allocation step, every actor of the merged application is bound to a tile in the architecture graph. As a result of binding the actors, the connections between the actors will be bound to either memory (in case both connected actors are bound to the same tile) or to a set of NoC links (in case the actors are bound to different tiles). Since the tile binding defines the computational load distribution and thus the power distribution within the MPSoC, it is the most important step in the thermal-aware mapping flow. A feasible tile binding binds all actors to a tile and all connections to a memory or interconnect link such that no storage, connection count or bandwidth limitation is violated. After a feasible tile binding is found, a static-order schedule is generated for each tile, defining the order of execution of the actors mapped on that tile. Note that the resulting resource allocation is not guaranteed to be able to meet the throughput constraint.

An extension of the heuristic-based resource allocation strategy introduced in [18] is used to find a feasible mapping that results in a power distribution close to the target power distribution. The binding algorithm is summarized in Algorithm 2. First, all actors are sorted on criticality in descending order. The criticality is calculated as a measure for the worst case. Next to approaching the target power distribution, there might also be other optimization targets, such as computational load balancing among the tiles, memory usage balancing or communication balancing/minimization. Function $cost(t, a)$, defined in Eqn. (1), assigns a cost to binding actor a to tile t, which is used in the binding algorithm. Constants $c_1, ..., c_6$ weight the costs of the different optimization criteria:

$$\begin{aligned} cost(t, a) = {} & c_1 * P(t, a) + c_2 * M(t, a) \\ & + c_3 * C(t, a) + c_4 * L(t, a) \\ & + c_5 * PDT(t, a) + c_6 * PDS(t, a) \end{aligned} \tag{1}$$

- $P(t, a) \in [0, 1]$: the normalized processor load when binding actor a to tile t;
- $M(t, a) \in [0, 1]$: the ratio of allocated memory when a is bound to t;
- $C(t, a) \in [0, 1]$: the ratio of allocated connections on tile t when a is bound to t;
- $L(t, a) \in [0, 1]$: the normalized average latency of all connections from/to a when a is bound to t;
- $PRT(t, a) \in [0, 1]$: a normalized cost for the power ratio of tile t when a is bound to t
  $$PRT(t, a) = c \cdot (r_t / R_t)$$
  where r_t is the estimated power ratio of tile t when binding a to t and R_t is the target power ratio of tile t. c is a normalizing constant to scale the cost to $[0, 1]$ for all tiles.
- $PRS(t, a) \in [0, 1]$: a normalized cost for the power ratio of the tile stack s containing tile t when mapping a to t.
  $$PRS(t, a) = c \cdot (r_s / R_s)$$
  where r_s is the estimated power ratio of stack s when binding a to t and R_s is the target power ratio of stack s. c is a normalizing constant to scale the cost to $[0, 1]$ for all stacks. A tile stack is a set containing all tiles that are in the same horizontal position at different layers. The target power ratio of a stack is defined by the sum of the ratios of the tiles it contains.

Note that there are two terms related to the power distribution: $PRT(t, a)$ and $PRS(t, a)$. PRT represents the deviation from the original target power ratios of the individual tiles. Since we observed that there is a large thermal correlation

ALGORITHM 2: Tile binding algorithm

```
Input: tiles $T$, actors $A$, connections $C$.
Output: Feasible resource allocation.
// Find an initial binding
Sort all actors $a \in A$ on criticality, descending;
for all sorted actors $a \in A$ do
    Sort all tiles $t \in T$ on $cost(t, a)$, ascending (see Eqn. (1));
    for all sorted tiles $t \in T$ do
        if binding actor $a$ to tile $t$ is feasible then
            Bind actor $a$ to tile $t$;
            Bind connections to/from $a$;
            break;
        end
    end
    if actor $a$ not bound then
        return "Unable to find feasible binding";
    end
end

// Try to improve the binding
Sort all actors $a \in A$ on criticality, ascending;
for all sorted actors $a \in A$ do
    Unbind actor $a$;
    Unbind connections to/from $a$;
    Sort all tiles $t \in T$ on $cost(t, a)$, ascending (see (1));
    for all sorted tiles $t \in T$ do
        if binding actor $a$ to tile $t$ is feasible then
            Bind actor $a$ to tile $t$;
            Bind all related connections;
            break;
        end
    end
end
```

between vertically adjacent tiles, it makes sense to also take the target power ratios of stacks of tiles into account to, which is captured by the PRS term. If the power distribution among stacks would not be included, deviations from the target power distribution in the vertical direction would result in the same cost as deviations in the horizontal direction, leading to worse results in a thermal sense. The tile power ratios resulting from a candidate binding can be calculated since the active and idle powers of all tiles are known, along with the execution time of every actor on every possible tile.

Throughput Computation: When a feasible resource allocation has been found, the maximum throughput of the mapped application has to be calculated in order to validate the throughput constraint. This is done by modeling the mapped application as a binding-aware SDFG, and performing a state-space exploration by simulating the self-timed execution of the graph [16]. The throughput is calculated as soon as a recurrent state is found during the execution.

Utilisation Minimization: It is possible that the maximum throughput of the mapped application is higher than the throughput constraint. From an energy and temperature perspective, it makes sense to slow down the execution as long as the throughput constraint is satisfied. This is done in the utilisation minimization step. It is assumed that every processor contains a TDMA system, in which a time slice can be reserved during which the processor is idle. To slow down execution, an idle time slice is inserted in the TDMA schedule of every active processor. The appropriate sizes of the idle time slices are determined by performing a binary search and recalculating the throughput after every step. The search is terminated once the actual throughput is not more than 10% above the throughput constraint.

V. Experimental Results

The performance of the proposed approach is tested by applying it to a set of synthetic benchmark applications as well as a set of real-life multimedia applications.

A. Experimental Setup

Benchmark applications: To evaluate the performance of the thermal-aware mapping approach, a set consisting of 4 application graphs is generated. Every application consists of 8 actors with random (Gaussian distributed) execution times and storage requirements. To evaluate the effect of the weights in the tile binding cost function, multiple mappings are generated for each set of applications, based on different cost function weights. To eliminate effects of the random generator, 3 sets of applications are generated. Next to the synthetic benchmark application set, a real-life application set consisting of four independent H.263 encoders (5 actors each) with a throughput constraint of 60 frames per second is constructed.

Target 3D MPSoC: The sets of benchmark applications are mapped on a tile-based 3D MPSoC consisting of 3 layers of 2$\times$2 identical tiles. Each tile consists of a processor, memory and network interface, all modeled as a single block as depicted in Figure 4. The active power of each tile is set to $1.5W$, the idle power is set to 10% of the active power. For the placement of blocks in a tile, two different tile floorplans are used, such that the processor blocks do not overlap. Tile floorplan 1 is used on the bottom and top layer, while floorplan 2 is used on the middle layer to avoid stacking all processor blocks exactly on top of each other, since the power density is highest in that block. The heatsink is connected (via a heat spreader) to the bottom layer. The other active (power dissipating) layers are thinned down to 50μm. Between two active layers, a 10μm thin layer containing thermal interface material (TIM) is modeled. The most important physical properties of the 3D IC model are listed in Table I. In the center of the NI block of every tile, a bundle of 8$\times$9 TSVs is placed. For the interconnect, a hybrid NoC-Bus design is assumed, consisting of a regular NoC in the horizontal plane and a multi-drop shared bus for vertical communication [6]. In this setup, every tile is assumed to have its own NoC switch, and every stack of tiles contains a shared bus. In the architecture graph, communication links are modeled as point-to-point connections. The latencies of all possible tile-to-tile connections are calculated based on the delay of the shortest path between the tiles. A hop in the horizontal plane is modeled as a delay of 2 time-units, a hop in the vertical direction as 1 time-unit.

Temperature simulation: For every mapping, an execution trace of $0.5s$ is generated. The execution patterns are periodic with a period much shorter than $0.5s$, making longer simulations obsolete. From the execution trace and the architecture

TABLE I. PHYSICAL PROPERTIES & HOTSPOT PARAMETERS

Parameter	Value
Tile size $[mm]$	2×2
Silicon thermal conductance $[W/(m \cdot K)]$	150
Silicon specific heat $[J/(m^3 \cdot K)]$	$1.75 \cdot 10^6$
TIM thermal conductance $[W/(m \cdot K)]$	4
TIM specific heat $[J/(m^3 \cdot K)]$	$4 \cdot 10^6$
TSV thermal conductance $[W/(m \cdot K)]$	300
TSV specific heat $[J/(m^3 \cdot K)]$	$3.5 \cdot 10^6$
TSV diameter $[\mu m]$	10
TSV pitch $[\mu m]$	20
Bottom layer thickness $[\mu m]$	200
Non-bottom layer thickness $[\mu m]$	50
TIM layer thickness $[\mu m]$	10
Convection resistance to ambient $[K/W]$	3.0
Heatsink side/thickness $[mm]$	$14 \times 14 \times 10$
Heatsink th. conductance $[W/(m \cdot K)]$	400
Heatsink specific heat $[J/(m^3 \cdot K)]$	$3.55 \cdot 10^6$
Ambient temperature$[K]$	300

specification, power traces are derived for every block. The power traces are used in the modified HotSpot 5.02 thermal simulator to simulate the temperature with a grid resolution of 32×32 and a temporal resolution of $10\mu s$. First, a steady state simulation is performed to find a representative initial temperature distribution. Next, the transient temperature simulation is performed. Table I lists the most important HotSpot parameters.

Interconnect energy computation: Since the interconnect energy consumption can be a significant part of the total energy consumption [19], it is also interesting to investigate the communication intensity and interconnect energy consumption resulting from different mappings. Note that the computational energy consumption will be close to identical for all mappings, since a homogeneous architecture is considered and every application is slowed down to match the throughput constraint.

The interconnect consumes energy to facilitate communication between the titles and consumed energy is also referred to as communication energy. Between two tiles, communication has to take place when actors (tasks) mapped on them need to communicate with each other. The communication energy depends on the data volume and the relative locations of the communicating task (actor) pair. For each communicating task pair mapped to tile i & tile j and connected by edge e, the communication energy is estimated by the product of the number of transferred bits ($nrTokens[e] \times tokenSize[e]$) and the energy required to transfer one bit between tiles i & j ($E_{bit}(i,j)$), as defined in Equation 2. The value of $E_{bit}(i,j)$ is calculated based on the energy required for horizontal link traversals, vertical link traversals and the energy consumed in routers between tiles i & j, as shown in Equation 3. Vertical interconnects are implemented as shared buses, so no intermediate routers are involved when traversing multiple layers. Therefore, hops in the vertical direction will increase the total number of routings by just 1, independent of the number of vertical hops. The total communication energy is estimated by summing over all communicating task pairs (edges).

$$E_{comm}(e) = (nrTokens[e] \times tokenSize[e]) \times E_{bit}(i,j) \quad (2)$$

$$\begin{aligned} E_{bit}(i,j) = (&E_{bit}^{horizontal} \times hops_{horizontal}(i,j)) \\ &+(E_{bit}^{vertical} \times hops_{vertical}(i,j)) \\ &+(E_{bit}^{router} \times numOfRouters(i,j)) \end{aligned} \quad (3)$$

TABLE II. TARGET POWER DISTRIBUTION FOR THE CONSIDERED 3D IC

	Tile position in the horizontal plane			
	(0,0)	(0,1)	(1,0)	(1,1)
Top layer	2.8%	2.8%	2.7%	2.4%
Middle layer	7.5%	7.6%	7.4%	6.9%
Bottom layer	15.0%	15.3%	15.0%	14.7%

In our 3D IC model, the horizontal link energy per bit, $E_{bit}^{horizontal}$, is taken as 0.127 pJ, which is estimated from [6]. The vertical link energy per bit, $E_{bit}^{vertical}$, is determined by the used TSVs, and is therefore referred to as E_{bit}^{TSV}. E_{bit}^{TSV} is estimated to be 9.56$\times 10^{-3}$ pJ [20]. For a horizontal link length of 2mm, the per bit router energy E_{bit}^{router} is approximately 70% of $E_{bit}^{horizontal}$ [21]. E_{bit}^{TSV} is only 7.5% of $E_{bit}^{horizontal}$, providing substantial space for communication energy optimization by exploiting the low link energy in the vertical direction. However, using more vertical links may result in a higher peak temperature due to increased power density because of mapping communicating tasks on stacked tiles.

B. Thermal Profiling Results

The target power distribution obtained by running the thermal profiling algorithm on the considered 3D IC is given in Table II. It is observed that the power is almost completely balanced in the horizontal plane, indicating that the power dissipated in a stack of tiles is minimized. In the static power distribution that minimizes the peak temperature on this specific 3D IC, about 60% of the total chip power is dissipated in the bottom layer. About 29% is dissipated in the middle layer, and the remaining 11% is dissipated in the top layer. Further experiments show that this distribution mainly depends on the power distribution within the tiles, the layer thickness and the inter-layer bonds. Although the thermal properties of the heatsink have a large impact on the average chip temperature, the optimal power distribution is almost independent from it.

C. Benchmark Application Results

To evaluate the performance of the resource allocation strategy, 5 different combinations of tile binding cost function weights are evaluated, corresponding to different optimization objectives:

1) **Load balancing (LB)**: $(c_1, c_2, c_3, c_4, c_5, c_6) = (1,0,0,0,0,0)$. Balance the computational load as much as possible.
2) **Communication latency minimization (CLM)**: $(c_1, c_2, c_3, c_4, c_5, c_6) = (0,0,0,1,0,0)$. Minimize the interconnect latency.
3) **Load balancing + latency minimization (LB+CLM)**: $(c_1, c_2, c_3, c_4, c_5, c_6) = (1,0,0,1,0,0)$. Combine computational load balancing and latency minimization with equal weights.
4) **Power balancing by stack (PBS)**: $(c_1, c_2, c_3, c_4, c_5, c_6) = (0,0,0,0,0,1)$. The power

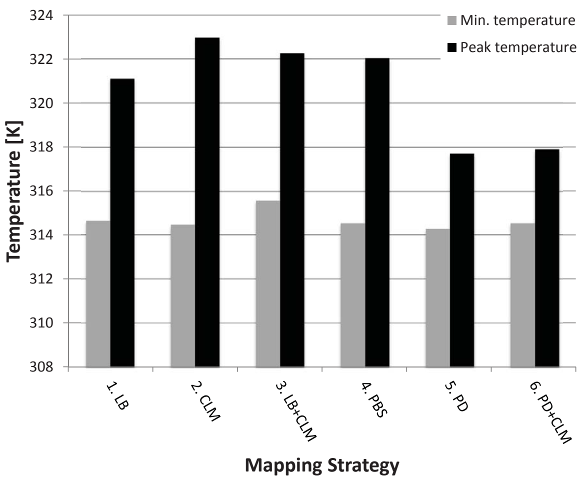

Fig. 8. Minimum and peak temperature resulting from mappings with different optimization objectives.

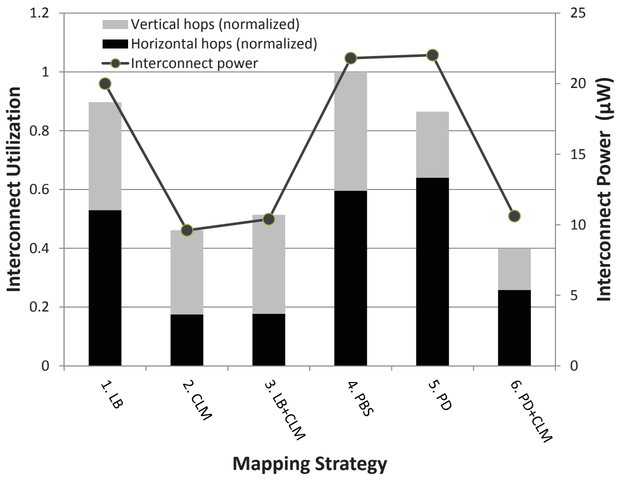

Fig. 9. Average normalized horizontal/vertical interconnect utilization and interconnect power.

ratios of all tile stacks are set equal, causing power balancing in the horizontal plane.

5) **Optimize power distribution (PD)**: $(c_1, c_2, c_3, c_4, c_5, c_6) = (0, 0, 0, 0, 1, 1)$. Optimize for a power distribution close to the target distribution.
6) **Optimize power distribution + latency minimization (PD+CLM)**: $(c_1, c_2, c_3, c_4, c_5, c_6) = (0, 0, 0, 1, 1, 1)$. Combine power distribution optimization with latency minimization.

The optimal combination of weights depends on the specific optimization objective for the considered system. Here, our goal is to evaluate the performance of the approach for some typical optimization objectives.

Temperature: Figure 8 shows the lowest and highest observed temperatures resulting from mapping the benchmark application set using the different optimization objectives. All mappings result in a throughput within 10% above the constraint. The results are averaged over the 3 application sets to remove effects of the random generator.

It is clear that trying to minimize the communication latency alone (scenario 2) results in the highest peak temperature. This is due to the fact that vertically adjacent tiles have smaller communication delays, which results in communicating actors being mapped on vertically adjacent tiles. This can cause a power imbalance in the horizontal plane, explaining the increased temperature. Including the target power ratio terms leads to a peak temperature decrease of $5.3K$ compared to the latency minimization case, and $3.4K$ compared to the computational load balancing case. It is clear that only balancing the load in the horizontal plane (scenario 4) does not result in the minimum temperature.

Interconnect usage and energy consumption: Figure 9 depicts the average normalized number of bit hops, as well as the average interconnect power consumption for different optimization criteria. A bit hop is defined as 1 bit of data that is transferred 1 hop through the NoC. The interconnect power is estimated as the average communication energy per second. It can be observed that including the latency cost term results in a significant decrease in interconnect utilization. This can be explained by the observation that the latency cost term assigns high costs to mappings in which communicating actors are mapped on tiles that are far apart in the 3D NoC. In scenario 6, the interconnect utilization is roughly halved compared to scenario 5, with almost no increase in peak temperature (Figure 8).

The average interconnect power depends on the usage of the horizontal and vertical interconnect links, as can be seen in Figure 9. A couple of observations can be made from Figure 9. First, the total power consumption is mainly governed by the usage of horizontal links, as they consume much more power than the vertical links. Second, the power consumption for the *CLM* and *LB+CLM* strategies are low due to more usage of vertical links than horizontal links, but they show high peak temperatures (Figure 8). This indicates that using more vertical links instead of horizontal links facilitates lower interconnect power consumption at the cost of a higher peak temperature due to more power stacking within the chip. Thus, a trade-off exists between minimizing interconnect energy consumption and peak temperature. Third, the *PD+CLM* strategy shows almost the same power consumption as the *CLM* and *LB+CLM* strategies, along with lower interconnect utilization and peak temperature. This indicates that the *PD+CLM* strategy results in a good balance between peak temperature, interconnect utilization and interconnect energy consumption. The absolute interconnect energy reduction highly depends on the communication intensity of the applications as well as the NoC type and technology.

D. Case-study for real-life applications

To test the applicability of our approach for real-life applications, 4 independent H.263 encoder applications are mapped on the 3D MPSoC using our mapping flow. Two different mapping strategies are applied: *LB+CLM* and *PD+CLM*, as introduced earlier. The *LB+CLM* strategy tries to balance the computational load while minimizing the interconnect latency,

TABLE III. Average interconnect power consumption, minimum and maximum temperature for 4 independent H.263 encoder applications.

	LB+CLM	PD+CLM
Interconnect power consumption (μW)	166.62	85.10
Minimum temperature (K)	310.75	310.15
Maximum temperature (K)	317.05	314.25

whereas the *PD+CLM* strategy aims at optimizing the power distribution while also minimizing interconnect latency.

Table III shows the average interconnect power consumption, minimum and maximum temperature when the mapping strategies *LB+CLM* and *PD+CLM* are employed. Strategy *PD+CLM* outperforms strategy *LB+CLM* for all the performance figures, i.e. it results in a lower interconnect power consumption, minimum temperature and maximum temperature. Minimizing the communication latency (*CLM*) results in a significant reduction in interconnect utilization and interconnect power consumption (Figure 9), making it an important optimization criterion to be considered. The results in Table III indicate that in addition to minimizing the communication latency, optimizing the power distribution (*PD*) is a better choice than balancing the computational load (*LB*).

VI. Conclusions and Future Work

We proposed a flexible and fast approach for thermal-aware mapping of throughput-constrained streaming applications on 3D MPSoCs. As compared to the load balancing case, the proposed approach reduces the peak temperature by 7% (in °C) and interconnect energy consumption by 47% for a set of benchmark applications on a 3-layer IC, while meeting all storage and throughput constraints. We showed that the approach can also be used in combination with other optimization criteria, such as interconnect utilization minimization. The average running time of the total flow is 20 minutes, with about 90% being spent in the thermal profiling step. The running time of the resource allocation & throughput validation step highly depends on the size and complexity of the application graph.

In future work, we plan to use more fine grained power models to increase the modeling accuracy. We also plan to develop a smarter utilization minimization method to take more advantage of slack, for example by minimizing the instantaneous power dissipation in the entire MPSoC. Finally, we want to implement a mechanism to automatically alter the cost function weights if the combination of weights results in a mapping that cannot meet the performance requirements.

References

[1] J. U. Knickerbocker, C. S. Patel, P. S. Andry, C. K. Tsang, L. P. Buchwalter, E. J. Sprogis, H. Gan, R. R. Horton, R. J. Polastre, S. L. Wright *et al.*, “3-d silicon integration and silicon packaging technology using silicon through-vias,” *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 8, pp. 1718–1725, 2006.

[2] B. S. Feero and P. P. Pande, “Networks-on-chip in a three-dimensional environment: A performance evaluation,” *Computers, IEEE Transactions on*, vol. 58, no. 1, pp. 32–45, 2009.

[3] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, “Toward dark silicon in servers,” *Micro, IEEE*, vol. 31, no. 4, pp. 6–15, 2011.

[4] A. K. Coskun, J. L. Ayala, D. Atienza, T. S. Rosing, and Y. Leblebici, “Dynamic thermal management in 3d multicore architectures,” in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE’09.* IEEE, 2009, pp. 1410–1415.

[5] E. A. Lee and D. G. Messerschmitt, “Static scheduling of synchronous data flow programs for digital signal processing,” *Computers, IEEE Transactions on*, vol. 100, no. 1, pp. 24–35, 1987.

[6] Y. Cheng, L. Zhang, Y. Han, and X. Li, “Thermal-constrained task allocation for interconnect energy reduction in 3-d homogeneous mpsocs,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 2, pp. 239–249, 2013.

[7] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, “Temperature-aware microarchitecture: Modeling and implementation,” *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 1, no. 1, pp. 94–125, 2004.

[8] X. Zhou, J. Yang, Y. Xu, Y. Zhang, and J. Zhao, “Thermal-aware task scheduling for 3d multicore processors,” *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 1, pp. 60–71, 2010.

[9] C. Addo-Quaye, “Thermal-aware mapping and placement for 3-d noc designs,” in *SOC Conference, 2005. Proceedings. IEEE International.* IEEE, 2005, pp. 25–28.

[10] T. Chantem, X. Hu, and R. P. Dick, “Temperature-aware scheduling and assignment for hard real-time applications on mpsocs,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 19, no. 10, pp. 1884–1897, 2011.

[11] C. Sun, L. Shang, and R. P. Dick, “Three-dimensional multiprocessor system-on-chip thermal optimization,” in *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2007 5th IEEE/ACM/IFIP International Conference on.* IEEE, 2007, pp. 117–122.

[12] L. Thiele, L. Schor, I. Bacivarov, and H. Yang, “Predictability for timing and temperature in multiprocessor system-on-chip platforms,” *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 1s, pp. 48:1–48:25, Mar. 2013. [Online]. Available: http://doi.acm.org/10.1145/2435227.2435244

[13] V. Nookala, D. J. Lilja, and S. S. Sapatnekar, “Temperature-aware floorplanning of microarchitecture blocks with ipc-power dependence modeling and transient analysis,” in *Proceedings of the 2006 international symposium on Low power electronics and design.* ACM, 2006, pp. 298–303.

[14] P. Zhou, Y. Ma, Z. Li, R. P. Dick, L. Shang, H. Zhou, X. Hong, and Q. Zhou, “3d-staf: scalable temperature and leakage aware floorplanning for three-dimensional integrated circuits,” in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on.* IEEE, 2007, pp. 590–597.

[15] M. Pathak and S. K. Lim, “Thermal-aware steiner routing for 3d stacked ics,” in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on.* IEEE, 2007, pp. 205–211.

[16] A. H. Ghamarian, M. Geilen, S. Stuijk, T. Basten, A. Moonen, M. Bekooij, B. Theelen, and M. R. Mousavi, “Throughput analysis of synchronous data flow graphs,” in *Application of Concurrency to System Design, 2006. ACSD 2006. Sixth International Conference on.* IEEE, 2006, pp. 25–36.

[17] S. Stuijk, M. Geilen, and T. Basten, “Sdfˆ3: Sdf for free,” in *Application of Concurrency to System Design, 2006. ACSD 2006. Sixth International Conference on.* IEEE, 2006, pp. 276–278.

[18] S. Stuijk, T. Basten, M. Geilen, and H. Corporaal, “Multiprocessor resource allocation for throughput-constrained synchronous dataflow graphs,” in *Design Automation Conference, 2007. DAC’07. 44th ACM/IEEE.* IEEE, 2007, pp. 777–782.

[19] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-ghz mesh interconnect for a teraflops processor,” *Micro, IEEE*, vol. 27, no. 5, pp. 51–61, 2007.

[20] (2010) International technology roadmap for semiconductors. [Online]. Available: http://www.itrs.net/reports.html

[21] S. Bhat, “Energy models for network-on-chip components,” MSc. thesis, Eindhoven University of Technology, 2005.