TAPHS: Thermal-Aware Unified Physical-Level and High-Level Synthesis

Zhenyu (Peter) Gu† Yonghong Yang‡ Jia Wang† Robert P. Dick† Li Shang‡

†EECS Department
Northwestern University
Evanston, IL 60208, U.S.A.
{zgu646, jwa112, dickrp}@ece.northwestern.edu

‡ECE Department
Queen's University
Kingston, ON K7L 3N6, Canada
4yy6@qlink.queensu.ca, li.shang@queensu.ca

Abstract— Thermal effects are becoming increasingly important during integrated circuit design. Thermal characteristics influence reliability, power consumption, cooling costs, and performance. It is necessary to consider thermal effects during all levels of the design process, from the architectural level to the physical level. However, design-time temperature prediction requires access to block placement, wire models, power profile, and a chip-package thermal model. Thermal-aware design and synthesis necessarily couple architectural-level design decisions (e.g., scheduling) with physical design (e.g., floorplanning) and modeling (e.g., wire and thermal modeling).

This article proposes an efficient and accurate thermal-aware floorplanning high-level synthesis system that makes use of integrated high-level and physical-level thermal optimization techniques. Voltage islands are automatically generated via novel slack distribution and voltage partitioning algorithms in order to reduce the design's power consumption and peak temperature. A new thermal-aware floorplanning technique is proposed to balance chip thermal profile, thereby further reducing peak temperature. The proposed system was used to synthesize a number of benchmarks, yielding numerous designs that trade off peak temperature, integrated circuit area, and power consumption. The proposed techniques reduces peak temperature by 12.5 °C on average. When used to minimize peak temperature with a fixed area, peak temperature reductions are common. Under a constraint on peak temperature, integrated circuit area is reduced by 9.9% on average.

1. Introduction

Increasing performance requirements and system integration are dramatically increasing integrated circuit (IC) power density and, therefore, cooling costs. Energy consumption and thermal issues are now central to the design of ICs, including both high-end instruction processors in general-purpose computers and application-specific integrated circuits (ASICs) in low-cost portable electronic consumer devices. Peak local temperature influences the reliability, packaging costs, cooling costs, bulk, and performance of IC. Many of these considerations are particularly important for portable devices.

Increasing IC power consumption raises average and peak temperatures. Temperature variations and hot spots account for over 50% of electronic failures [1], most of which are due to electromigration, hot carrier effects, thermal stress, and oxide thermal breakdown. Power and thermal variation can also lead to significant timing uncertainty, requiring more conservative timing margins, thereby reducing performance. Designers must frequently trade off other design metrics, such as performance, area, and cooling costs, to meet tight thermal constraints. The interaction of power and thermal constraints with other design metrics further increases system complexity. As projected by the International Technology Roadmap for Semiconductors (ITRS) [2], further process scaling will be bounded by power consumption and heat dissipation below 65 nm: it is critical to address energy and thermal issues during IC design to meet the urgent needs of the semiconductor industry and enable future technology scaling.

This work is supported in part by the NSF under awards CNS-0347941 and CCR-0238484, and in part by an NSERC Discovery Grant #388694-01.

Thermal problems cannot be well solved at any single level of the design process. Thermal characterization requires detailed physical information, including an IC floorplan and power profile as well as interconnect and chip-package thermal models. Thermal optimization requires a unified high-level and physical-level design flow. At the architectural level, reducing supply voltage can reduce IC power consumption, hence the temperature, while at the physical level, peak temperature can further be reduced by modifying the IC floorplan to balance the thermal profile. Furthermore, the evaluation and optimization of the tradeoff between IC temperature and other design metrics, such as performance, area, and cooling cost, requires a comprehensive architectural-level and physical-level infrastructure.

Incremental synthesis is a promising design technique that may be used to unify high-level synthesis and physical design. It improves the quality of results by maintaining important physical-level properties across consecutive physical design changes, many of which are triggered by architectural changes [3–5]. Moreover, it dramatically improves synthesis time by reusing and building upon high-quality, previous physical design solutions that required a huge amount of time and effort to produce.

This paper presents a thermal-aware, floorplanning, incremental high-level synthesis system called TAPHS. The proposed incremental synthesis techniques rapidly determine the impacts of architectural changes on floorplan-dependent characteristics and concurrently optimize IC thermal profile, area, and energy consumption under performance constraints.

2. Related work

In this section, we survey related work in the two main research areas in which TAPHS is rooted: (1) high-level and physical-level co-synthesis and (2) thermal-aware analysis and design.

As a result of technology scaling, it is becoming increasingly important to consider the physical design impacts of high-level design decisions. This requires floorplanning and interconnect estimation at the highest levels of design. A few researchers have previously considered incremental floorplanning [6] and the impact of incorporating loosely-coupled constructive floorplanners within high-level synthesis [7], [8]. Other researchers subsequently used incremental floorplanning and synthesis [4] to tightly couple high-level and physical synthesis [3].

Recent studies on thermal issue focus on analysis and optimization. A number of thermal analysis approaches that try to efficiently model chip-package designs have been proposed [9–13]. Thermal and thermal-reliability issues are becoming increasingly important for IC interconnection networks due to their influence on electronmigration and stress migration voiding. Recent studies [14], [15] have proposed numerical and analytical modeling techniques to characterize the thermal profile of on-chip interconnect layers. Thermal issues have also been considered during chip cell-level placement [16], [17], three-dimensional IC floorplanning [18], and high-level synthesis resource sharing [19].

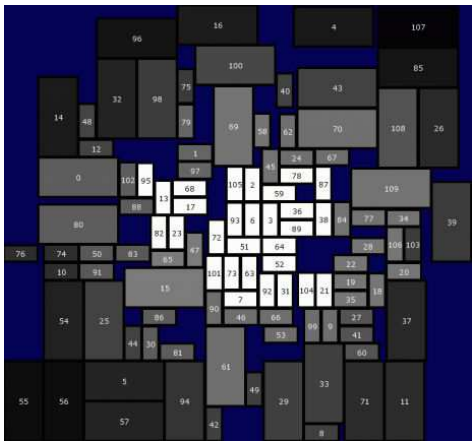

Figure 1. **Post-synthesis thermal profile without voltage islands.**

3. Motivating example

In this section, we use an example IC design to demonstrate the challenges of thermal optimization in high-level synthesis. Figure 1 shows an IC floorplan produced by an integrated high-level synthesis and floorplanning algorithm. In this figure, the numbered rectangles are functional units, e.g., adders, multipliers, dividers, or registers. Using thermal analysis, as described in Section 8, the IC thermal profile is determined. The temperature of each functional unit is indicated by its brightness: brighter functional units are hotter. 85 °C is a typical thermal emergency threshold to ensure reliable operation. In this example, functional units temperatures higher than 85 °C are white. 29 of the functional units are operating at dangerously high temperatures: this chip is likely to suffer from failure caused by thermal-related reliability problems, e.g., electromigration. Note that producing the detailed chip thermal profile in Figure 1 requires detailed physical information, i.e., a floorplan, a power profile, and a chip-package thermal model. Therefore, stand-alone high-level synthesis algorithms have no means of detecting, let alone correcting, thermal crises.

High-level synthesis provides numerous thermal optimization opportunities. Reducing supply voltage reduces power consumption, hence temperature, but may also impair performance. Recent work on voltage islands has proposed operating different regions of an IC at different voltages. Figure 2 illustrates the floorplan of an IC using voltage islands. In this design, functional units are assigned to contiguous voltage islands with different supply voltages. The brightnesses of the thick functional unit boundaries indicate their voltages. In this example, three voltage islands are used. As in Figure 1, functional units violating the 85 °C temperature constraint are white.

A comparison of Figures 1 and 2 indicates that voltage islands can dramatically improve thermal conditions. The number of functional units with temperatures above the thermal constraint decreased from 29 to 19. However, as shown in Figure 2, localized hot spots still exist. The remaining hot spots are primarily the result of local concentrations in power density. Although changing the assignment of operations to functional units may improve local temperatures in some circumstances, in practice we found that the vast majority of thermal problems in designs already using voltage islands could only be resolved with thermal-aware physical design techniques, especially for designs with tight deadlines.

Our study suggests a rich set of high-level and physical-level thermal optimization techniques, including multiple operating voltages, appropriate scheduling, and thermal-aware floorplanning. Many of the techniques to optimize IC thermal properties also impact other design metrics such as area and power consumption. We have considered the side effects of a number of techniques, proposing those that allow improvements to thermal properties while maintaining good area, performance, and power consumption.

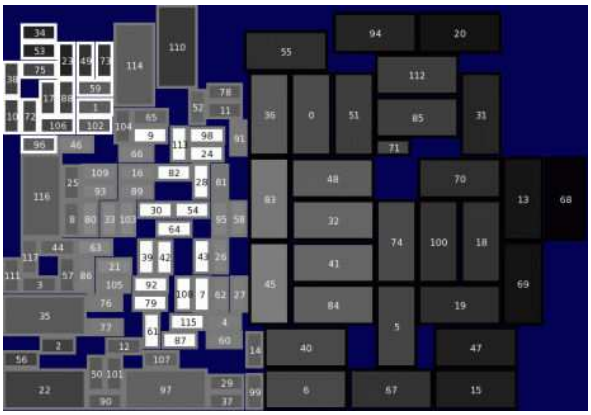

Figure 2. **Post-synthesis thermal profile with voltage islands.**

Using voltage islands has a significant impact on chip area and performance as well as increasing the complexity of floorplanning. Voltage islands require the addition of voltage converters and delivery circuits, as well as on-chip level shifters to support communication among functional units in different voltage islands. Moreover, reduced supply voltage requires a longer clock period to compensate for reduced switching speeds. In order to use voltage islands, a synthesis algorithm must wisely choose the island for each functional unit, appropriately allocate timing slack to allow scheduling, and generate floorplans in which functional units in the same voltage island are contiguous. This tightly couples the architectural and physical levels of design.

Thermal-aware floorplanning is challenging because it must trade off multiple conflicting objectives: peak temperature, IC area, and power consumption. Adjusting functional unit positions to balance power density, thereby reducing peak IC temperature, may increase chip area. Moreover, if the high-power functional units are also high-activity functional units, as would be expected as a result of resource sharing, they are also likely to frequently communicate with other functional units. In general, high-connectivity functional units hosting operations on critical timing paths ought to be placed near each other to minimize interconnect delay and power consumption. However, doing so can have the side effect of increasing peak IC temperature.

Facing these design challenges, a high-quality thermal-aware synthesis system must incorporate thermal optimization techniques into a unified high-level and physical level design flow, as well as striking wise tradeoffs among conflicting design goals.

4. Overview of TAPHS

In this section, we give an overview of TAPHS: our incremental thermal-aware physical and high-level synthesis system. TAPHS considers the thermal impact of both logic and interconnect power dissipation. It automatically generates voltage islands and schedules operations to reduce IC power consumption and peak temperature. In addition, it does thermal-aware floorplan optimization.

Figure 3 illustrates the main algorithms used in TAPHS. First, the control and data flow graph is simulated with typical input traces in order to profile each operation and data transfer edge. The profile information, an RTL design library, floorplanner, and thermal model are used to evaluate the IC temperature profile, power, area, and performance. Slack distribution, voltage clustering, and voltage island aware floorplanning are used to generate voltage islands for use in the initial solution: a fully parallel implementation. There are two loops within the high-level synthesis algorithm. In the outer loop, the clock period of the design is iteratively changed from the minimum to maximum potentially feasible values. Incremental rescheduling, resource sharing, resource splitting (i.e., the opposite of resource sharing), and slack distribution are used to generate valid solutions. In the inner loop, back-tracking iterative improvement is used to op-

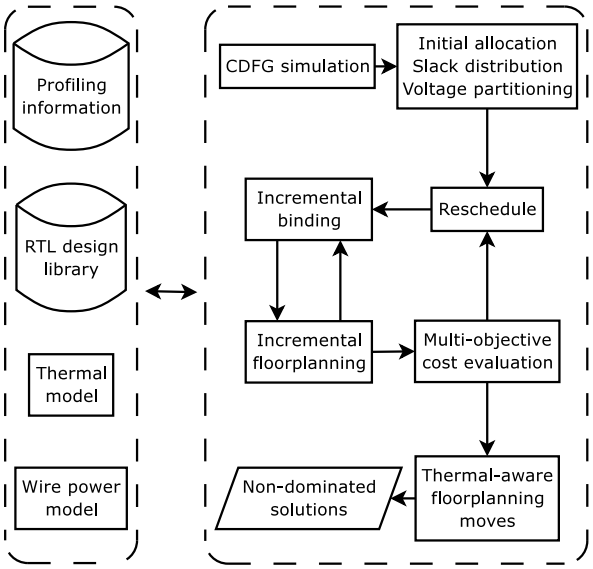

Figure 3. Incremental high-level synthesis algorithm

timize the RTL architecture, considering multiple objectives, e.g., peak temperature, area, and power consumption. A *dominated* solution is inferior to some other previously encountered solution in all costs. Non-dominated solutions are preserved in a solution cache, from which the designer may choose based upon the desired trade-offs among costs.

A high-quality thermal-aware incremental floorplan was developed and incorporated into TAPHS. Each time the high-level synthesis algorithm needs thermal and physical information to guide its moves, it extracts that information from the current, incrementally generated, floorplan. In addition, costs derived from the floorplan are used to guide high-level synthesis moves. By using incremental floorplanning, closer interaction between high-level synthesis and physical design is possible, i.e., the high-level synthesis algorithm may determine the impact of potential changes to binding upon physical attributes such as IC thermal profile, area, and interconnect energy consumption.

5. Slack distribution

To allow voltage scaling, it is necessary to appropriately distribute scheduling slack among operations. *Slack* is the difference between latest and earliest start time. Determining whether it is possible, and desirable, to assign an operation to a lower-voltage functional unit is not possible based on as soon as possible (ASAP) operation start times. TAPHS redistributes slack among operations in order to support more energy-optimal assignment of functional units to voltage islands.

Assume that control and data flow graphs have been partitioned into same-slack paths, as described later in this subsection. Given a single path composed of sequential operations, the slack distribution problem is equivalent to deciding the execution time of each operation such that energy consumption is minimized under a hard constraint on path execution time. We shall use the following variables and constants: D is the bound on path execution time; p is the set of all operations on the path; d_i is the delay of an operation's functional unit; v_i is the voltage of an operation's functional unit; V_t is the threshold voltage constant; K is an execution time constant; E is the total path energy consumption; e_i is the energy required for an operation; C_i is the switched capacitance constant of an operation's functional unit; and α is the alpha power law constant [20].

$$d_i = \frac{K v_i}{(v_i - V_t)^{\alpha}} \quad \text{subject to the constraint } D \geq \sum_{i \in p} d_i \tag{1}$$

However, V_t is small and a very low value of v will generally imply an unacceptable path delay that will be prevented by the constraint on line 1. Therfore, we may assume V_t is small, thus

$$d_i \simeq \frac{K v_i}{v_i^{\alpha}} \quad \text{and} \quad v_i = \left(\frac{d_i}{K}\right)^{\frac{1}{1-\alpha}} \tag{2}$$

$$e_i = C_i v_i^2 = C_i \left(\frac{d_i}{K}\right)^{\frac{2}{1-\alpha}} \quad \text{and} \quad E = \sum_{i \in p} C_i \left(\frac{d_i}{K_i}\right)^{\frac{2}{1-\alpha}} \tag{3}$$

$$\min_{\substack{\forall i \in p \\ v_i}} \quad \sum_{i \in p} C_i \left(\frac{d_i}{K_i}\right)^{\frac{2}{1-\alpha}} \tag{4}$$

Note that a decrease in v_i implies a decrease in e_i, which implies an increase in d_i. Therefore, for minimal E, $D = \sum_{i \in p} d_i$. Consider the delay and energy trade-off for an arbitrary pair of operations:

$$d_{12} = d_1 + d_2 \quad \text{and} \quad e_{12} = e_1 + e_2 \tag{5}$$

$$e_{12} = \frac{C_1}{K_1{}^{\frac{2}{1-\alpha}}} (d_1)^{\frac{2}{1-\alpha}} + \frac{C_2}{K_2{}^{\frac{2}{1-\alpha}}} (d_{12} - d_1)^{\frac{2}{1-\alpha}} \tag{6}$$

Take the derivative of e_{12} with respect to d_1, set to zero, and solve to find d_2/d_1 for minimal E.

$$\frac{d_2}{d_1} = \left(\frac{\frac{C_1}{K_1{}^{\frac{2}{1-\alpha}}}}{\frac{C_2}{K_2{}^{\frac{2}{1-\alpha}}}}\right)^{\frac{1-\alpha}{1+\alpha}} \tag{7}$$

This optimal delay ratio for two operations may be used to compute the optimal delay ratio for an arbitrary pair of operations. These ratios can be scaled by a dynamically computed value, N, to ensure that the constraint on line 4 is honored.

$$N = \sum_{i \in p} \frac{d_i}{d_1} \tag{8}$$

$$\forall_{i \in p}\, d_i = \frac{D}{N} \left(\frac{\frac{C_1}{K_1{}^{\frac{2}{1-\alpha}}}}{\frac{C_i}{K_i{}^{\frac{2}{1-\alpha}}}}\right)^{\frac{1-\alpha}{1+\alpha}} \quad \text{or} \quad \frac{D}{N} \sqrt[3]{\frac{C_i K_i{}^2}{C_1 K_1{}^2}} \quad \text{for } \alpha = 2 \tag{9}$$

Equations 6 and 9 yield the optimal time, d_i, to dedicate to each operation. By granting slack to each operation in the path such that its time is proportional to its time share, we allow the voltage island generation algorithm the opportunity to assign functional units to voltage islands such that energy consumption may be minimized under a hard constraint on path execution time (please see Section 6).

Thus far, we have discussed individual operation paths. However, it is necessary for TAPHS to determine slack distributions along numerous paths in arbitrary directed acyclic graphs of operations. Assigning time shares eventually has the effect of (temporarily) fixing operation start times. These start times may influence the earliest start times and latest finish times of operations on other paths; in order to avoid deadline violations, slack distribution is conducted on operation paths in order of increasing path slack. In order to generate paths, a modified depth-first search is conducted on a graph in which each vertex is an operation labeled with its slack and each edge is a data dependency. Vertex children are visited in increasing order of slack, thereby guaranteeing that vertices on multiple paths will be included in minimal-slack paths.

As shown in Algorithm 1, starting from the minimal-slack path, TAPHS incrementally assigns extra clock cycles to operations. At each step, it locates the operation, j, for which the current allocated time, t_j, differs most from d_j (Step 7) and assigns it an additional

Algorithm 1 Slack distribution procedure

1: Compute all operation slacks
2: Group operations into same-slack paths, P
3: Sort paths P in order of increasing slack
4: **for all** $p \in P$ **do**
5: **while** slack remains on p **do**
6: $\forall_{i \in p}\, t_i$ is the time assigned to operation i
7: Operation $i = \arg\min_j \sqrt[3]{\frac{C_j K_j^2}{C_1 K_1^2}} \cdot \frac{D}{N} - t_j$ by Equation 9
8: Assign one additional clock cycle to operation i
9: **end while**
10: Recompute all operation slacks
11: **end for**

clock cycle (Step 8). It is guaranteed that this will not result in deadline violations on other paths because slack distribution is carried out on paths in order of increasing slack. Therefore, slack distribution on a given path is prevented from delaying any node so much that slack becomes negative on other paths on which the node lies. After slack sharing is done for a given path, the slacks of all nodes are recomputed and slack distribution proceeds for the next path.

6. Voltage partitioning

TAPHS uses on-chip voltage islands to optimize IC thermal profiles and energy consumption. On-chip voltage islands are generated in two stages. *Voltage partitioning* classifies functional units into different voltage levels to maximize overall power and energy savings hence potential IC temperature reduction. *Voltage island generation* is then conducted via incremental floorplanning to produce and optimize voltage islands.

In this section, we focus on voltage partitioning under two design constraints. First, reducing supply voltage increases circuit propagation delay. Hence, the minimal supply voltage of a functional unit is constrained by its available time slack. Second, increasing the number of on-chip voltage levels introduces significant overhead to off-chip and on-chip power supply and delivery circuits. Therefore, only a limited, design-dependent, number of voltage levels should be generated.

In this work, we propose an efficient voltage partitioning algorithm. It conducts optimal voltage allocation and assignment to maximize overall power savings and strike judicious trade-offs among different design metrics.

Motivating example

We next present an example to illustrate the voltage partitioning problem. Consider a circuit design with five functional units as shown in Fig. 4. For each functional unit, FU_i, the minimal allowed supply voltage, $V_{FU_i}^{min}$, is uniquely determined by the ratio of its time slack to its propagation delay under the initial (maximum) supply voltage. In a voltage partition Ψ_i^S with S clusters, to satisfy the deadline constraints of functional units, for each cluster, $\psi_j = \{FU_{j1}, \ldots, FU_{jn}\}$, $\psi_j \in \Psi_i^S$, its supply voltage, V_{ψ_j}, is greater than or equal to $max\{V_{FU_{jt}}^{min}\}_{t=1,\ldots,n}$, i.e., the minimal supply voltage of the functional unit with the lowest slack-delay ratio inside this cluster. Consider the voltage partitioning shown in Fig. 4(a). This partitioning contains two voltage clusters, $\psi_1 = \{FU_1, FU_2\}$ and $\psi_2 = \{FU_3, FU_4, FU_5\}$. The supply voltage of ψ_1, V_{ψ_1} is 1 V, which is the minimal allowed supply voltage of FU_2. The supply voltage of ψ_2, V_{ψ_2} is 2 V, which is the minimal allowed supply voltage of FU_5.

Fig. 4(c) shows the energy consumptions of different voltage partitions, which are derived using a linear scan along the functional unit list. This list is sorted in order of increasing slack-delay ratios (or minimal allowed supply voltages) of functional units. This figure shows that, using linear scan, the energy curve is not monotonic, implying that an algorithm with $O(N)$ time complexity is necessary to find a single voltage partitioning cut with minimal energy consumption.

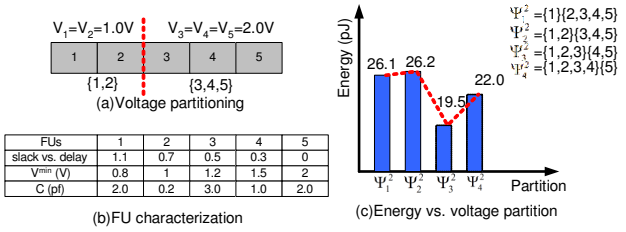

Figure 4. **Voltage partitioning example.**

We now define the optimal voltage partitioning problem.

Problem Definition *Given N functional units,* $\{FU_1, \ldots, FU_n\}$*, and an input M, find an optimal voltage partition,* Ψ_{opt}^M*, containing M voltage clusters,* $\{\psi_{opt\,j}\}_{j=1,\ldots,M}$*, such that its energy consumption,* $E(\Psi_{opt}^M) \le E(\Psi_i^M), \forall \Psi_i^M$*, in which* $E(\Psi_{opt}^M) = \sum_{l=1}^{N} C_l \times V_{\psi_{opt\,j}}^2$*.* C_l *is the capacitance of* $FU_l, FU_l \in \psi_{opt\,j}, j = 1, \ldots, M$.

For each functional unit, FU_i, to satisfy its deadline constraint, its minimal allowed supply voltage, $V_{FU_i}^{min}$, is uniquely determined by the ratio of its slack time to its propagation delay under the initial (maximum) supply voltage. Then, for each cluster, $\psi_j = \{FU_{j1}, \ldots, FU_{jn}\}$, $\psi_j \in \Psi_i^M$, its supply voltage, $V_{\psi_j} \ge max\{V_{FU_{jt}}^{min}\}_{t=1,\ldots,n}$, i.e., the minimal supply voltage of the functional unit with the lowest slack to delay ratio inside this cluster.

An optimal voltage partitioning is derived using the following approach. Functional units are first sorted by their slack to propagation delay ratios. Then, linear scans along the sorted functional unit list determine the optimal partitioning. Note that the energy saving curve is not monotonic, implying that an algorithm with $O(N)$ time complexity is required to find an energy-optimal voltage partitioning. For M partitions, the time complexity of this algorithm is $O(N^M)$.

An optimal voltage partitioning algorithm of $O(N^2)$ complexity

We introduce an optimal voltage partitioning algorithm of $O(N^2)$ time complexity. Its pseudo-code is shown in Algorithm 2, which is described in a recursive form. *Partition*() has five input/output parameters. $*FU\_list$ points to the sorted functional unit list. *Start* and *End* designate the sub-list: the portion of the original list that needs to be partitioned. Initially, $Start = 0$ and $End = N$ denote voltage partitioning targets on the whole sorted list. M defines the targeted number of partition cuts. *OptTable* stores intermediate optimal partitions of sub-lists.

Partition() is invoked recursively when $M > 1$ (line 1–4). For each sub-partitioning (M cuts) on a sub-list (from *Start* to *End*), the optimal solution is derived using a linear scan to examine the M^{th} cut from *Start* to *End*, which is combined with the optimal solution of the sub-partitioning ($M-1$ cuts) on its sub-list (from i to *End*). When $M = 1$, the algorithm uses a linear scan to find the optimal cut in the targeted sub-list (line 7).

Lemma 1 *In an optimal partition* Ψ_{opt}^M *with M voltage clusters,* $\{\psi_{opt,1}, \ldots, \psi_{opt,M}\}$*, and* $V_{\psi_{opt,1}} \ge V_{\psi_{opt,2}} \ge \cdots \ge V_{\psi_{opt,M}}$*, then* $V_{\psi_{opt,i}} \le V_{FU_j}^{min} \le V_{\psi_{opt,i-1}}, \forall FU_j \in \psi_{opt,i}$.

Lemma 1 implies that the optimal partitioning can be found by partitioning the sorted functional unit list. This lemma guarantees the optimality of the algorithm: it uses a linear scan to explore all the possible partitioning combinations of the sorted list, including the optimal solution. Using linear scan to find the optimal M partitions on a sorted list with N functional units, the computational complexity is $O(N^M)$. To improve computation efficiency, we use a data structure, called *OptTable*, to store optimal sub-partitions. The time complexity of partitioning M results from a linear scan of the M^{th} cut multiplied by the time complexity of finding the optimal $M-1$ partitions, which only requires a linear search in *OptTable* table (line 5) with complexity $O(N)$. In total, there are M recursive

Algorithm 2 *Partition*$(*FU\_list, Start, End, M, *OptTable)$

1: **if** $M > 1$ **then**
2: $C \leftarrow 0$
3: **for** $(i \leftarrow Start; i \leq End; i++)$ **do**
4: $Partition(*FU\_list, i, End, M--, *OptTable)$
5: $E^M_{M^{th}=i} \leftarrow C \times (V^{min}_{FU_{i-1}})^2 + OptTable[M-1][i]$
6: $C+ = C_{FU_i}$
7: **end for**
8: $E^M_{opt}(Start, End) \leftarrow min\{E^M_{M^{th}=i}\}_{i \leftarrow Start,\ldots,End}$
9: $cut^M_{opt}(Start, End) \leftarrow i$ *if* $E^M_{M^{th} \leftarrow i} = E^M_{opt}(Start, End)$
10: $OptTable[M][Start] \leftarrow pair(E^M_{opt}(Start, End), cut^M_{opt}(i, End))$
11: **else**
12: $Linear\_Scan(*FU\_list, End, \&E^1_{opt}(Start, End), \&cut^1_{opt}(Start, End))$
13: $OptTable[1][Start] \leftarrow pair(E^1_{opt}(Start, End), cut^1_{opt}(Start, End))$
14: **end if**

layers. Since M is much smaller than N, the overall time complexity of this optimal voltage partitioning algorithm is $O(N^2)$.

7. Thermal-aware floorplanning

In order to support thermal-aware, incremental, unified high-level and physical-level optimization, it was necessary to incorporate a high-quality, incremental floorplanner within TAPHS. New algorithms were developed and incorporated into this floorplanner to directly support physical-level thermal optimization and indirectly support architectural-level thermal optimization.

The floorplanner within TAPHS is based on the Adjacent Constraint Graph (ACG) representation [21]. An ACG is a constraint graph with exactly one geometric relationship between every pair of modules. ACGs have invariant structural properties that allow the number of edges in the graph to be bounded. Operations on ACGs have straightforward meanings in physical space and change graph topology locally; they require few, if any, global changes. The operations of removing and splitting modules are designed to reflect high-level operation to functional unit binding decisions. To obtain the physical position of each module, packing based on longest path computation is employed. Simulated annealing is used to obtain an initial floorplan. A weighted sum of the area and the interconnect power consumption is calculated for use as the floorplanner cost function, i.e.,

$$A + w \sum_{e \in E} C_e D_e \tag{10}$$

where A is the area, w is the power consumption weight, E is the set of all wires, e is an interconnect wire, C_e is the unit-length switched capacitance for the data transfer along e, and D_e is the length of e, which is calculated as Manhattan distance between the two modules connected by the wire. Using this cost function, the floorplanner optimizes the interconnect power consumption, interconnect delay, and area. The resulting floorplan will be improved during the subsequent incremental floorplanning high-level synthesis moves. Therefore, the number of simulated annealing iterations is bounded to reduce synthesis time.

After each high-level synthesis move, the previous floorplan is modified by removing or splitting a module. The modules and switched capacitances are updated based upon the impact of these merges and splits. The floorplan is then re-optimized with a greedy iterative improvement algorithm using the same cost function as the simulated annealing algorithm. There are two reasons to use a greedy algorithm during this stage of synthesis: (1) re-optimization requires fewer global changes and less hill climbing and (2) perturbations resulting from high temperatures may disrupt high-quality floorplans.

After determining the best binding across all the possible clock frequencies, another simulated annealing floorplanning run is used for that binding. This final floorplanning stage occurs only once for every synthesis run. Therefore, it is acceptable to use a slower, but higher-quality, annealing schedule than those in the inner loop of high-level synthesis, thereby improving IC area and interconnect power consumption.

During the annealing schedule, we use a constant cooling factor, r, i.e.,

$$T^+ = r \times T \tag{11}$$

where T is the current temperature and $T+$ is the temperature during the next iteration. The number of the perturbations for the initial floorplanning run, the floorplanning for each clock frequency, and the final floorplanning are related as follows: $1:2:20$. The number of perturbations per round for the greedy iterative improvement algorithm is the same as that for final floorplanning run.

7.1. Voltage island implementation in floorplanning

As described in Section 6, voltage island generation was introduced into the high-level synthesis system in order to improve thermal profiles and reduce energy consumption. Therefore, the floorplanner must attempt to keep functional units assigned to the same voltage level contiguous in order to minimize the need for level converters and simplify power distribution. The floorplanner must still honor the elements in the original cost function shown in Equation 10. Pair-wise weighted edges were added between all pairs of functional units operating at the same voltage, yielding the following updated cost function:

$$n\sqrt{A} + 2n \sum_{v \in V} L_v + \sum_{e \in E} C_e D_e \tag{12}$$

where A is the area, n is the number of functional units, V is the set of all functional unit pairs sharing the same voltage, v is a pair of functional units sharing the same voltage, L_v is the separation between a pair of functional units sharing the same voltage, E is the set of all interconnects, e is an interconnect line, C_e is the unit-length switched capacitance for the data transfer along e (zero in the case of no communication), and D_e is the length of e. This approach generates contiguous voltage islands, as well as optimizing area and interconnect power consumption.

Figure 2, described in Section 3, shows an example of the results produced by this floorplanning algorithm. TAPHS rapidly generated this result using only pair-wise edges for functional unit clustering, i.e., hierarchical floorplanning was not required. Note that functional units operating at the same voltage are contiguous. In some cases, keeping functional units within voltage islands contiguous and minimizing wire length results in a slight area penalty. This is to be expected, regardless of the quality of a floorplanner, because it is rare for a minimal-area solution to maintain contiguous voltage levels and minimal interconnect power consumption. During incremental improvement, operation merging (functional unit resource sharing) combines functional units with other compatible functional units, always merging from the lower-voltage functional unit to the higher-voltage functional unit in order to honor performance constraints (please see Section 4).

7.2. Thermal-aware swap operation

As explained in Section 3, hot spots may occur because a number of functional units with high power densities are physically close to each other. Such concentrations are natural. It is common for high-activity, high-power functional units to frequently communicate with other high-activity functional units. This causes the floorplanner to position the functional units near each other in order to reduce interconnect power consumption. However, in some cases, the objectives of minimizing average power consumption and minimizing peak temperature conflict with each other.

We propose a thermal-aware swap operation that exchanges hot, generally high power density, functional units with cool, generally low power density, functional units within the same voltage island. This heuristic sorts compatible functional units in a voltage island in order of increasing temperature. The positions of the highest and lowest temperature functional units are the exchanged, after which the exchanged functional unit positions are locked and the operation is repeated. The thermal-aware functional unit swapping heuristic halts after some proportion of the functional units have been moved. In practice, a proportion of 1/3 allowed a significant reduction in peak temperature for most examples.

8. Experimental results

In this section, we present experimental results for the TAPHS thermal-aware high-level synthesis system, including the thermal optimization techniques described in Sections 5, 6, and 7. The circuits described in this section were synthesized using a register transfer level (RTL) design library based on the TSMC 0.18 μm process. The experiments were conducted on AMD Athlon-based Linux workstations with 512 MB–1 GB of random access memory. All IC synthesis runs required less than 1,195 s of CPU time.

8.1. Thermal model

As mentioned in Section 4, thermal modeling and analysis are used in the inner loop of the optimization flow to provide direct guidance for thermal optimization. Therefore, in order to determine the thermal profile of our system, we integrated our original work, a compact chip-package thermal model [13], into TAPHS. The thermal model has been validated against FEMLAB [22], an accurate but slow commercial finite-element based simulator, with less than 2.5% estimation error on the Kelvin scale. In the following experiments, each chip design is attached to a copper heat sink using forced air-cooling. We model two thermally conductive paths: heat dissipates from the silicon die through the cooling package to the ambient environment and through the package to the printed circuit board. We use an ambient temperature of 45 °C and a silicon thickness of 200 μm. In high-end microprocessor systems more than 80% of heat is dissipated through the first conductive path. In portable consumer electronic devices, due to the tight cooling budget and limited cooling space, the impact of the secondary conductive path becomes significant.

8.2. Benchmarks

We used TAPHS to synthesize 13 synthesis benchmarks. *Chemical* and *IIR77* are infinite impulse response (IIR) filters used in industry. *DCT_IJPEG* is the Independent JPEG Group's implementation of discrete cosine transform (DCT). *DCT_Wang* and *DCT_Lee* are DCT algorithms named after their inventors. All DCT algorithms work on 8×8 pixel of arrays. *Elliptic*, an elliptic wave filter, comes from the NCSU CBL high-level synthesis benchmark suite [23]. *Jacobi* is the Jacobi iterative algorithm for solving a fourth order linear system. *WDF* is a finite impulse response (FIR) wave digital filter. The largest benchmark, Jacobi, has 24 multiplications, 8 divisions, 8 additions, and 16 subtractions. In addition, we generated two CDFGs using a pseudo-random graph generator [24]. Random100 has 20 additions, 15 subtractions, and 19 multiplications. Random200 has 39 additions, 44 subtractions, and 36 multiplications. The same sample periods (deadlines) were used for the benchmarks when evaluating each synthesis technique.

8.3. Multiobjective results

Table 1 shows the results of doing full multiobjective optimization of peak temperature, area, and energy consumption. In total, we compared 13 benchmarks. For each benchmark, the table shows non-dominated solutions produced by TAPHS. Due to space con-

Table 1. Comparison of non-dominated (multiobjective) results

	No voltage islands			Voltage islands			Thermal FP	
Example	Peak T (°C)	Area (%)	Power (W)	Peak T (°C)	Area (%)	Power (W)	Peak T (°C)	Power (W)
chemical	123.4	116.6	2.18	98.0	142.4	1.60	93.6	1.48
	123.6	112.0	2.18	100.4	121.7	1.62	96.2	1.51
	123.7	109.3	2.18	103.3	112.7	1.59	99.7	1.50
	128.6	112.9	2.24	110.3	95.0	1.62	105.3	1.53
dct_dif	79.0	87.9	0.85	67.3	92.5	0.60	65.6	0.55
	79.7	78.6	0.83	67.6	81.5	0.58	66.1	0.54
	80.3	83.7	0.85	69.8	83.4	0.61	67.4	0.57
	80.1	81.4	0.84	69.3	74.9	0.57	67.6	0.53
	81.7	80.7	0.86	69.9	80.0	0.60	68.4	0.56
	82.9	76.0	0.87	71.3	78.8	0.63	68.5	0.57
	84.5	68.8	0.87	71.4	75.8	0.62	68.7	0.57
dct_ijpeg	126.0	118.2	2.44	113.6	117.6	1.99	106.9	1.79
	129.4	107.2	2.39	115.8	114.9	2.03	107.4	1.81
	129.5	104.5	2.41	118.6	99.9	2.00	110.6	1.80
	130.6	104.7	2.40	118.9	102.0	2.03	111.2	1.82
	140.9	93.6	2.45	122.8	92.0	2.04	113.3	1.84
dct_lee	71.5	98.9	0.79	63.7	106.4	0.59	62.3	0.54
	71.8	95.6	0.79	65.5	119.2	0.61	62.4	0.55
	71.9	99.9	0.79	64.6	106.3	0.59	63.1	0.54
	75.0	87.8	0.80	65.2	100.4	0.60	63.6	0.55
	73.8	91.9	0.79	65.8	107.9	0.60	64.0	0.54
	73.7	91.7	0.79	66.8	106.4	0.62	65.0	0.57
	73.9	102.0	0.82	68.1	112.3	0.64	65.8	0.57
	73.6	102.0	0.81	68.7	101.2	0.64	66.3	0.58
dct_wang	70.7	101.3	0.70	59.8	109.8	0.42	57.6	0.39
	68.2	97.5	0.68	59.1	116.0	0.43	57.9	0.40
	68.5	108.1	0.68	60.1	108.0	0.42	58.1	0.39
	70.4	89.1	0.70	59.8	102.8	0.44	58.3	0.41
	71.3	100.5	0.69	61.1	100.1	0.45	59.3	0.42
	70.3	101.0	0.70	61.2	113.0	0.45	59.6	0.42
	72.0	85.1	0.72	63.1	109.8	0.48	60.7	0.43
	72.4	77.4	0.70	65.2	91.8	0.47	61.4	0.42
	72.0	88.9	0.72	66.3	90.8	0.48	63.6	0.43
	70.8	86.6	0.70	66.7	78.2	0.47	64.5	0.43
elliptic	136.8	105.5	2.55	111.6	122.6	2.04	108.0	1.93
iir77	94.5	105.0	1.57	73.7	119.7	0.94	72.0	0.89
	97.7	93.1	1.56	74.6	115.7	0.94	72.9	0.90
	99.0	93.1	1.57	76.5	94.9	0.96	75.2	0.92
jacobi	54.2	64.4	0.25	51.8	81.5	0.20	51.3	0.19
	53.9	65.5	0.25	52.1	77.7	0.20	51.5	0.19
	53.8	63.2	0.24	52.9	69.2	0.21	52.1	0.20
	54.9	59.4	0.25	52.5	69.4	0.21	52.2	0.20
	54.2	60.0	0.24	53.1	65.2	0.21	52.5	0.20
	54.4	66.0	0.25	53.1	61.7	0.21	52.6	0.20
	54.8	58.7	0.25	53.3	62.4	0.22	52.7	0.21
	54.6	58.0	0.25	53.6	64.5	0.22	53.3	0.21
	55.0	57.4	0.25	53.6	61.4	0.22	53.4	0.21
	55.5	52.9	0.25	54.5	61.5	0.22	53.5	0.20
pr1	98.0	104.0	1.49	82.5	106.1	1.10	80.3	1.02
	97.4	103.1	1.52	84.8	92.6	1.10	81.9	1.02
pr2	95.4	103.8	1.67	87.9	110.2	1.44	83.5	1.32
	97.3	89.2	1.68	87.5	100.6	1.45	83.6	1.33
	95.8	98.4	1.67	88.0	105.4	1.45	84.2	1.32
	99.3	91.2	1.68	88.4	98.4	1.44	84.6	1.32
	98.7	90.5	1.68	90.3	102.3	1.47	85.5	1.34
	99.9	79.8	1.71	91.7	83.0	1.47	86.6	1.34
	98.6	84.5	1.70	92.2	88.7	1.47	86.7	1.34
random100	71.6	100.0	0.85	66.0	98.8	0.63	63.6	0.57
	72.1	99.2	0.85	65.7	99.6	0.62	64.2	0.58
	72.7	99.7	0.86	67.6	85.1	0.67	64.6	0.62
	73.2	85.4	0.86	67.2	87.3	0.64	64.6	0.60
	74.1	91.5	0.86	66.5	92.3	0.63	64.7	0.58
	73.8	89.5	0.88	66.3	98.7	0.63	64.7	0.58
	73.7	94.5	0.88	68.6	87.2	0.66	65.3	0.61
	73.7	87.8	0.86	68.4	82.2	0.64	65.3	0.59
	75.0	89.1	0.88	69.4	86.5	0.68	65.3	0.62
	74.4	86.3	0.87	68.1	79.3	0.64	65.5	0.60
	73.9	87.0	0.85	68.9	80.4	0.66	66.1	0.61
	74.2	85.5	0.87	74.2	76.8	0.72	67.2	0.66
	76.5	84.1	0.87	73.5	62.9	0.73	69.8	0.66
	79.1	68.3	0.88	73.1	71.2	0.72	70.1	0.66
random200	90.8	90.2	1.77	81.4	112.0	1.37	76.2	1.20
	91.1	93.0	1.77	83.2	90.2	1.37	78.6	1.20
wdf	75.6	108.0	0.75	68.0	104.5	0.59	65.4	0.55
	74.8	96.9	0.73	67.8	101.8	0.59	67.0	0.55

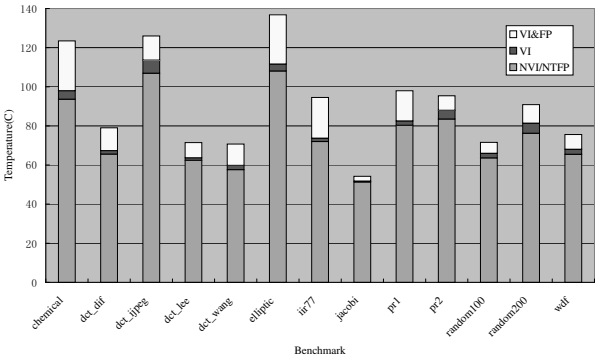

Figure 5. Peak temperature comparison

straints, we sorted the solutions for each problem in order of increasing peak temperature and uniformly eliminated all but seven solutions.

For each solution, the left column indicates the name of the benchmark. The next three columns show the peak temperatures, areas, and power consumptions of solutions produced without using voltage islands or thermal-aware floorplanning moves. Area is reported as a percentage of the area of the an initial solution without resource sharing or voltage islands. The floorplanner typically has an area efficiency ranging from 75%–90% for these benchmarks. From these columns, it should be clear that it is possible to trade off peak temperature for area as long as a thermal model is available during multiobjective synthesis. However, improving both objectives requires architectural-level and physical-level thermal optimization techniques.

The next three columns show the results produced using voltage islands, but without using thermal-aware floorplanning moves. From these columns, it is clear that voltage islands yield significant improvements in peak temperature, area, and power consumption. For example, the peak temperatures of the lowest peak temperature solutions to each problem were reduced by an average of 12.5 °C.

The next two columns show the results produced using both voltage islands and thermal-aware floorplaning moves. Note that the areas of these solution are the same as those without thermal-aware floorplanning moves. Combined with voltage islands, this technique allowed an average of 3.1 °C reduction in peak temperature.

Figure 5 shows only the lowest peak temperature for each benchmark after synthesis with voltage islands and thermal-aware floorplanning moves, with voltage islands but without thermal-aware floorplanning moves, and without voltage islands. As this figure indicates that both voltage islands and thermal-aware floorplanning moves can substantially reduce IC peak temperature, and that the relative contribution of each technique depends on the benchmark. In general, the best results were produced when these techniques were used together.

In addition, given the same area, TAPHS achieves lower peak temperatures for most benchmarks. For example, the peak temperature of *pr2* was reduced from 95.8 °C to 88.4 °C with the same area. Similar reduction were possible for *dct_dif*, *dct_ijpeg*, and *dct_lee*. In addition to reducing peak temperature, the proposed techniques can also be used to reduce area given a fixed peak temperature. When constraining temperature to the lowest temperature solution found without thermal optimization techniques, using voltage islands and thermal-aware floorplanning reduced area by, on average, 9.9%.

9. Conclusions

In this paper, we have described TAPHS, a thermal-aware high-level synthesis system that uses a tightly integrated thermal model and incremental floorplanner to optimize ICs peak temperatures, areas, and power consumptions, while meeting performance constraints. In order to optimize peak temperature, it was necessary to tightly integrate floorplanning, wire modeling, power profile generation, and chip-package thermal analysis with high-level synthesis. Experimental results indicate that TAPHS is able to trade off peak temperature, IC area, and power consumption. The proposed techniques allowed a reduction in peak temperature of 12.5 °C, on average. Peak temperature was also reduced under a fixed area constraint. Moreover, we have found that thermal optimization can allow significant improvements in IC area under temperature constraints. We conclude that it is important to incorporate thermal optimization in high-level synthesis to support continued increases in device and power density.

References

[1] L.-T. Yeh and R. C. Chu, *Thermal Management of Microelectronic Equipment: Heat Transfer Theory, Analysis Methods, and Design Practices*. New York, NY: ASME Press, 2002.

[2] International Technology Roadmap for Semiconductors, http://public.itrs.net.

[3] Z. P. Gu, *et al.*, "Incremental exploration of the combined physical and behavioral design space," in *Proc. Design Automation Conf.*, June 2005, pp. 208–213.

[4] O. Coudert, *et al.*, "Incremental CAD," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2000, pp. 236–244.

[5] J. Cong and M. Sarrafzadeh, "Incremental physical design," in *Proc. Int. Symp. Physical Design*, Apr. 2000.

[6] W. Choi and K. Bazargan, "Incremental placement for timing optimization," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003.

[7] L. Zhong and N. K. Jha, "Interconnect-aware high-level synthesis for low power," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2002, pp. 110–117.

[8] A. Stammermann, *et al.*, "Binding, allocation and floorplanning in low power high-level synthesis," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003.

[9] Y. Cheng, *et al.*, "ILLIADS-T: An electrothermal timing simulator for temperature-sensitive reliability diagnosis of CMOS VLSI chips," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 8, pp. 668–681, Aug. 1998.

[10] P. Li, *et al.*, "Efficient full-chip thermal modeling and analysis," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2004, pp. 319–326.

[11] H.-S. Wang, *et al.*, "Orion: A power-performance simulator for interconnection networks," in *Proc. Int. Symp. Microarchitecture*, Dec. 2002, pp. 294–305.

[12] K. Skadron, *et al.*, "Temperature-aware microarchitecture," in *Proc. Int. Symp. Computer Architecture*, June 2003, pp. 2–13.

[13] L. Shang, *et al.*, "Thermal modeling, characterization and management of on-chip networks," in *Proc. Int. Symp. Microarchitecture*, Dec. 2004, pp. 67–80.

[14] T.-Y. Chiang, K. Banerjee, and K. C. Saraswat, "Analytical thermal model for multilevel VLSI interconnects incorporating via effect," *IEEE Electron Device Letters*, vol. 23, no. 1, pp. 31–33, Jan. 2002.

[15] Z. Lu, *et al.*, "Interconnect lifetime prediction under dynamic stress for reliability-aware design," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2004, pp. 327–334.

[16] C. Tsai and S. Kang, "Cell-level placement for improving substrate thermal distribution," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 253–266, Feb. 2000.

[17] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force directed approach," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2003, pp. 86–89.

[18] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 2004, pp. 306–313.

[19] R. Mukherjee, S. O. Memik, and G. Memik, "Temperature-aware resource allocation and binding in high-level synthesis," in *Proc. Design Automation Conf.*, June 2005.

[20] K. A. Bowman, *et al.*, "A physical alpha-power law MOSFET model," *J. Solid-State Circuits*, vol. 34, pp. 1410–1414, Oct. 1999.

[21] H. Zhou and J. Wang, "ACG–Adjacent constraint graph for general floorplans," in *Proc. Int. Conf. Computer Design*, Oct. 2004.

[22] "COMSOL Multiphysics," http://www.comsol.com/products/multiphysics.

[23] "NCSU CBL high-level synthesis benchmark suite," www.cbl.ncsu.edu/benchmarks.

[24] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. Int. Wkshp. Hardware/Software Co-Design*, Mar. 1998, pp. 97–101.