An Integrated Approach to Thermal Management in High-Level Synthesis

Rajarshi Mukherjee, *Member, IEEE*, and Seda Ogrenci Memik, *Senior Member, IEEE*

***Abstract*—Thermal effects are becoming an important factor in the design of integrated circuits due to the adverse impact of temperature on performance, reliability, leakage, and chip packaging costs. Making all phases of the design flow aware of this physical phenomenon helps in reaching faster design closure. In this paper, we present an integrated approach to thermal management in architectural synthesis. Our synthesis flow combines temperature-aware scheduling and binding based on feedback from thermal simulation. We show that our flow is effective in preventing hotspot formation and creating an even thermal profile of the resources. Our integrated thermal management technique on average reduces the peak temperature of the resources by 7.34 °C when compared to a thermal unaware flow without increasing the number of resources across our set of benchmarks.**

***Index Terms*—Architectural synthesis, resource allocation, resource assignment, scheduling, temperature.**

I. Introduction

CONTINUOUS technology scaling will enable a billion transistor integrated circuits (ICs) by the end of this decade. All types of electronic systems are geared with increasing computational power and logic density resulting in significantly higher power dissipation. The power that is dissipated is converted into heat, which can be spatially nonuniform across the system. Localized heating occurs at a much faster pace than chip wide heating due to the slow rate of lateral heat propagation [1]. This results in uneven temperature distribution. Such high temperature regions are commonly referred to as "hotspots." Low power techniques do little to alleviate hotspot formation since they may not reduce power density in hotspots. Therefore, it is necessary to construct thermal aware design flows.

High temperature adversely impacts circuit performance. With increasing temperature, interconnect resistance increases and switching speed of transistors decreases. Slow devices coupled with slow interconnects in the vicinity of hot functional units can cause timing violation and ultimately lead to transient functional failure. Even if excessive heat does not lead to spontaneous damage, it accelerates electromigration, which can lead to permanent damage in the long run. In addition to the reliability concerns, the dependency of leakage power on temperature creates a challenge for low power design. As the share of leakage in total budgets increases, the shrinking share of dynamic power will actually lead to less actual power being utilized for performance. Leakage has an exponential dependence on temperature. Therefore, high temperatures can cause a large shift in the power budget allocations in power constrained embedded system designs.

Various chip packaging and cooling techniques such as heat sink, fan, etc. have been developed. Gunther *et al.* [2] have shown a nonlinear relationship between the cooling capabilities and the cost of the solution as power dissipation increases. This shows the importance of limiting the maximum temperature for efficient control of heating. Aside from thermal packaging, there are two approaches to thermal management: 1) runtime thermal management and 2) design planning and optimization during synthesis.

In this paper, we have focused on integrating thermal management into the architectural synthesis flow. Specifically, we have developed a synthesis flow where temperature-aware optimizations are tightly integrated into multiple stages spanning across scheduling and resource allocation and assignment. Resource allocation, assignment, and scheduling are main tasks within architectural synthesis, which are also referred to as high-level synthesis (HLS) collectively. Each of these steps is complex and often performed separately to maintain low complexity and a modular flow. Individual optimization steps may be unaware of polices and objectives pursued in other stages. Combining multiple design stages into the optimization process will produce significantly better solutions.

For various optimization problems such as power minimization, researchers have attempted to couple one stage into another by either iteratively changing scheduling, binding, and allocation based on feedback or trying to perform these tasks simultaneously. A common approach has been to incorporate feedback from the physical synthesis stage into HLS [3], [4]. Simultaneous scheduling and allocation [5], scheduling and binding [6], and scheduling and floorplanning [7] have been proposed. Prabhakaran *et al.* in [8] proposed simultaneous scheduling, binding, and floorplanning. More recently, the bus binding problem has been integrated with scheduling to minimize power dissipation of the buses [9], [10]. In general, integrating multiple design stages tightly towards the same goal is highly beneficial. Additionally, incorporating feedback from lower levels into higher levels of the design flow also leads to better solution quality.

To the best of our knowledge, a multistage integrated temperature optimization in the architectural synthesis stage has not

Manuscript received June 30, 2005; revised January 30, 2006 and June 22, 2006. This work was supported in part by the National Science Foundation under Career Award 0546305.

R. Mukherjee is with Synopsys, Inc., Mountain View, CA 94043 USA.

S. O. Memik is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208 USA (e-mail: seda@ece.northwestern.edu).

Digital Object Identifier 10.1109/TVLSI.2006.886408

been attempted before. In this paper, we present a novel integrated approach to thermal management, where scheduling and binding stages are tightly coupled iteratively and both stages receive feedback from post-floorplan thermal simulation.

Our specific contributions in this paper are as follows.

- We formulate the problem of integrated thermal management in high-level synthesis.
- We develop a thermal feedback driven iterative scheduling—binding algorithm, which minimizes the maximum temperature reached by the hottest resource without increasing the number of resources used.
- We evaluate the impact of our thermal management scheme on power consumption. We show that our scheme causes a small deviation from optimal switching power, however, helps improve leakage power. Hence, we demonstrate that significant peak temperature control can be achieved with virtually no overhead and some improvement in leakage is obtained in the process as well.

The remainder of this paper is organized as follows. Section II gives an overview of a related paper. In Section III, we discuss our integrated thermal management approach. Section IV presents our experimental flow and results. We conclude with a summary in Section V.

II. Related Work

Different techniques have been proposed to enable even thermal distribution on chips by optimizations in the physical synthesis stage. Recently, Cong *et al.* [11] introduced a thermal-driven floorplanning algorithm for 3-D ICs. A thermal driven multilevel routing has been proposed in [12]. A standard cell placement tool for even thermal distribution has been developed by Tsai and Kang [13]. Chu and Wong proposed the matrix synthesis approach for thermal placement [14]. Basu *et al.* introduced the electrothermal energy-delay-product optimization scheme to perform simultaneous optimization of supply and threshold voltages in CMOS circuits [15]. A methodology has been developed for making temperature and reliability aware power, performance, and cooling-cost tradeoffs in an IC [16].

Tools for modeling thermal effects for the design of high performance microprocessors have been developed [17]–[19]. Intel Pentium-4 processors use runtime thermal management via clock gating using realtime temperature sensing [2]. Techniques such as frequency and/or voltage scaling, sub-banking, etc., [20]–[22] have been investigated in the past. Skadron *et al.* [1] developed HotSpot—a thermal model for microprocessors. Shang *et al.* [23] proposed ThermalHerd, a distributive, collaborative runtime thermal management scheme for on-chip networks. Yang *et al.* [24] proposed temporally and spatially adaptive dynamic and steady-state fast thermal analysis algorithms for ICs.

A thermal-aware realtime task scheduling algorithm for embedded systems has been proposed by Hung *et al.* [25]. Their work presents a runtime management solution. Recently, binding algorithms for temperature constrained resource minimization and resource constrained temperature minimization have been proposed by Mukherjee *et al.* [26]. Their work addresses only the resource allocation and binding stage. Also, temperature evaluation is performed using analytical models that relate switching activity and associated expected heat dissipation. Starting from an optimal low-power binding solution, an iterative rebinding algorithm for uniform activity distribution was proposed by Mukherjee *et al.* [27]. The authors demonstrated the effectiveness of activity reassignment in reducing the peak temperature of the design. However, the metric for task reassignment was the switching power of the resources instead of temperature. Gu *et al.* [28] proposed forming voltage islands during HLS and floorplanning to alleviate hotspot formation.

In this paper, we present an integrated approach to thermal management in architectural synthesis. We integrated task scheduling, resource allocation, and assignment, and post-floorplan thermal simulation into a single thermal-aware framework. A thermal-aware floorplan is generated using HotFloorplan [29], which manages lateral heat propagation through floorplanning. The temperature profile is obtained using the HotSpot simulator [18], which takes into account the geometry and floorplan of the functional units during thermal simulation.

In summary, our contribution in this paper distinguishes itself from existing work in the following aspects. We perform the first attempt to the best of our knowledge to incorporate thermal-awareness into multiple stages of the architectural synthesis flow in an integrated fashion. Our optimization framework provides thermal awareness during synthesis time and delivers temperature control and power optimizations in the resulting system by construction as opposed to runtime management. Although physical design stages such as thermal-aware flooplanning is successful in minimizing the peak temperature, we will show that our integrated high level synthesis flow is indeed effective in making early impactfull decisions and further reducing the maximum temperature. However, note that this provides opportunities for various runtime techniques and our synthesis techniques to coexist and complement each other. Also, in contrast to existing synthesis efforts that only address the resource binding stage, we propose a multistage flow that is tightly integrated with physical information provided from a thermal simulator.

III. Thermal Management in Architectural Synthesis

In this section, we first present a motivational example that illustrates the impact of proactive thermal management during synthesis. Next, we give a detailed description of our thermal management scheme.

A. Motivation

Let us consider two flows—thermal unaware HLS (TU-HLS) and thermal aware HLS (TA-HLS). The TU flow could be optimized for design metrics such as timing, power, and area. We have used switching power optimization for TU-HLS.

TU-HLS does not care about the individual activity of modules but minimizes total power, which in most cases results in uneven distribution of activity and, hence, creates hotspots. Fig. 1 compares the temperature profiles generated by these two alternative flows. TA-HLS, on the other hand, decreases the peak temperature of the design and creates an even thermal profile for the functional units. In this particular example, we are comparing the temperatures of twenty functional units of the

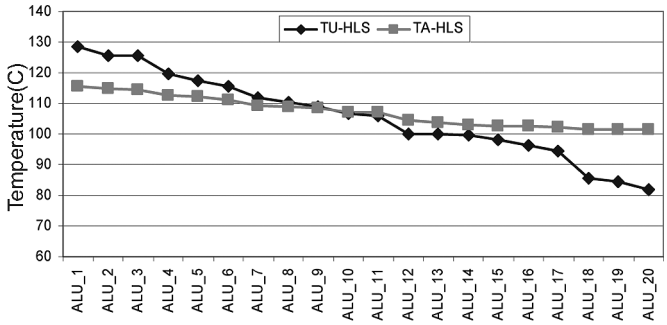

Fig. 1. Temperature profile across resources. The highest temperature reached by any resource has decreased from 128.4 °C to 115.4 °C.

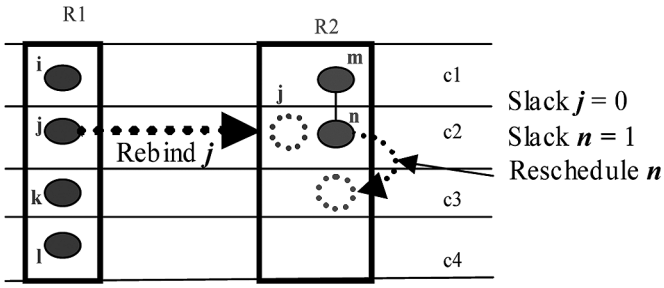

Fig. 2. Example of rescheduling and rebinding.

same type used in a design. ALU_1 reaches the highest temperature at 128.4 °C and ALU_20 reaches the lowest at 81.7 °C in TU-HLS. After using TA-HLS, we observe that the temperature of the hottest resource namely ALU_1 decreased to 115.4 ° C. Leakage has an exponential dependence on temperature. By effectively decreasing the maximum temperature of the hottest resource, TA-HLS helps decrease the leakage power by 6% at 100-nm technology node for this particular example.

B. Integrated Flow With Thermal Feedback

Typical stages in HLS are scheduling, resource allocation, and binding. Scheduling can be resource constrained latency optimized or latency constrained resource optimized. Scheduling dictates the compatibility of operations with respect to their assignments to functional units. Generally for scheduled data flow graphs (DFGs), this relation is captured in the form of a comparability graph. As a consequence, the topology of the comparability graph determines the solution space available to the subsequent binding stage. Binding determines the activity of functional units thereby affecting their thermal profile. Thus, scheduling indirectly controls the temperature of a functional unit as well. Therefore, efficient thermal management can be achieved by taking both of these stages and their intertwined relations into consideration. Any optimization that requires a change in either step will most likely require reorganization in the other. For instance, any rebinding might necessitate rescheduling. Let us consider a simple example shown in Fig. 2. Operations $[i, j, k,$ and $l]$ are bound to resources R1 and $[m, n]$ are bound to R2. The operations are scheduled at control steps c1 to c4. Op_j has zero slack. Op_j can be reassigned to R2 if only Op_n is rescheduled from control step c2 to c3. This is feasible since Op_n has one unit of time slack.

In our thermal management scheme, we address this tight relationship between scheduling and binding by employing an iterative rescheduling and rebinding approach. Furthermore, the iterations between the scheduling and binding stages are orchestrated by feedback obtained from a post-synthesis thermal simulation.

The temperature correlates well with the power dissipated by a resource. However, it also depends on the size and geometry of the resource as well as its relative location on the chip. Therefore, it is important to consider the actual temperature profile of the resources and provide feedback pertaining the physical considerations back into the synthesis flow. At the same time, any resynthesis must be "local," i.e., local changes are to be made such that most of the previous solution can be reused instead of creating a dramatically different solution for which the temperature profile may be totally different. It also must be "directed," i.e., there must be inexpensive means to identify the fewest operations for rescheduling and rebinding to reach a thermally optimized design.

Our thermal-aware flow works as follows. We start with a resource constrained latency minimized schedule. Then, we perform low power binding. Since there is a close relationship between power dissipation and temperature, such a power optimized binding solution is a good starting point. However, a power optimized solution alone is not sufficient to address the temperature control problem. Low-power binding assigns operations to functional units such that the total switching capacitance of the resources is minimized. In order to meet this objective, this technique can bind an uneven number of operations to different functional units. Even those functional units that have the same number of operations bound to them may have widely varying switching activities. As a result, some functional units can dissipate more power, thus, creating uneven power densities that may contribute to large variations in operating temperatures across the IC. Next, we create a thermal aware floorplan (by managing lateral heat spreading) and perform a thermal simulation of the resulting design that takes physical properties of the IC (such as thermal properties of the silicon), available cooling mechanisms (such as the properties of the heat sink) and the thermal interaction between different functional units on a floorplan into account. Based on the feedback from this thermal simulation, we iteratively perform operation specific rescheduling rebinding in order to gradually decrease the temperature of the hottest resource. Essentially, feedback from thermal simulation identifies a subset of operations that are to be evaluated for resynthesis.

During rebinding, we would ideally like to choose an operation to be reassigned to another resource such that the switching power of the hottest resource decreases maximally and the increase in switching power of the destination resource is minimum. Such a rebinding may not always be feasible due to conflicts. We will explain the conflicts in more detail in the next section. In order to remove such conflicts, we may have to reschedule operations, which creates a new comparability graph and makes rebinding feasible. Thus, thermal-aware flow integrates thermal feedback driven rescheduling and rebinding. These resynthesis decisions are local and the schedule and binding of a majority of the operations remain unchanged. We discuss the details of our algorithm in the next section.

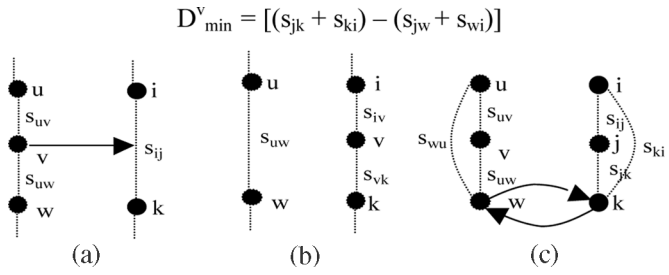

Fig. 3. (a), (b) Compatible insert: moves one operation onto a new resource. (c) Swap: exchanges two operations between two resources.

Lyuh *et al.* [10] presented a technique for rapidly generating new optimal network flow solution for a local change in a schedule. They first perform max-flow computation while retaining as much as the previous solution possible and then a min-cost computation step to refine the flow solution. For each reschedulable operation, the switching power after optimal binding is computed and the rescheduling move is accepted if it results in decreasing switching power. Trying all possible rescheduling moves and rebinding using the min-cost flow solution will not necessarily serve our purpose of thermal management. This approach does not guarantee to have a more uniform switching activity across resources. Instead, we choose to perform rescheduling of a selected set of operations only if suggested by the feedback derived from thermal simulation.

1) Thermal Feedback Driven Iterative Rescheduling and Rebinding Algorithm: In this section, we discuss our thermal feedback driven iterative rescheduling rebinding algorithm. The temperature profile of the functional units after the initial low-power binding is obtained by temperature simulation. From the temperature profile, we identify the resource $R^{\mathrm{T}}_{\max}$, which has the maximum temperature, and the resource $R^{\mathrm{T}}_{\min}$ with minimum temperature. Then, we try to migrate an operation from $R^{\mathrm{T}}_{\max}$ to $R^{\mathrm{T}}_{\min}$. We refer to this rebinding as a move. We will denote the start time of operation v as $T_s(v)$ and finish time as $T_f(v)$. We define three types of moves—*compatible insert, conflict insert, and swap*.

Operation v can be moved from $R^{\mathrm{T}}_{\max}$ to $R^{\mathrm{T}}_{\min}$ and placed between operations i and k that are assigned to $R^{\mathrm{T}}_{\min}$ already, if $T_f(i) \leq T_s(v) < T_f(v) \leq T_s(k)$. Operation v is then compatible with operations i and k and with resource $R^{\mathrm{T}}_{\min}$. Say, for operation w in $R^{\mathrm{T}}_{\max}$ there exists an operation k in $R^{\mathrm{T}}_{\min}$ such that $T_s(k) \leq T_s(w) \leq T_f(k)$ or $T_s(w) \leq T_s(k) \leq T_f(w)$. In this case operation w conflicts with k or $k = \mathrm{conflict}(w)$.

Let us consider a subset of operations $[u, v, w]$ bound to $R^{\mathrm{T}}_{\max}$ and $[i, k]$ bound on $R^{\mathrm{T}}_{\min}$ that execute consecutively. Let operation v be compatible with operations i and k, and be moved to $R^{\mathrm{T}}_{\min}$. This type of a move is called a compatible insert and is shown in Fig. 3(a) and (b). The decrease in switching capacitance of $R^{\mathrm{T}}_{\max}$ due to the removal of v is $D^{v}_{\max}$, where

$$D^{v}_{\max} = [(s_{\mathrm{uv}} + s_{\mathrm{vw}}) - s_{\mathrm{uw}}].$$

The increase in switching capacitance of $R^{\mathrm{T}}_{\min}$ due to insertion of v is $I^{v}_{\min}$, where

$$I^{v}_{\min} = [(s_{\mathrm{iu}} + s_{\mathrm{vk}}) - s_{\mathrm{ik}}].$$

If operations w and k have a conflict, w can be moved to $R^{\mathrm{T}}_{\min}$ if it is compatible with other operations on $R^{\mathrm{T}}_{\min}$ and similarly k can be simultaneously moved to $R^{\mathrm{T}}_{\max}$ if compatible with other operations on $R^{\mathrm{T}}_{\max}$. This type of move is called swap and is shown in Fig. 3(c). Note that the min-cost flow formulation is optimal in minimizing the total switched capacitance in a single iteration of a DFG, i.e., the intra-iteration switched capacitance. However, the optimality condition does not hold for cyclic execution of a DFG, i.e., inter-iteration switched capacitance is not minimized. In our algorithm, we consider optimization for intra- as well as inter-iteration switched capacitance. We disallow any swap move if it increases the intra-iteration switched capacitance, since it would increase the switching capacitance of both the resources. We only allow swaps if it decreases the inter-iteration switching capacitance. In Fig. 3(c) s_{wu} and ski denote the inter-iteration switching capacitance of $R^{\mathrm{T}}_{\max}$ and $R^{\mathrm{T}}_{\min}$, respectively. The decrease in switching capacitance of $R^{\mathrm{T}}_{\max}$ due to swapping operations w and k is $D^{v}_{\max}$, where

$$D^{v}_{\max} = [(s_{\mathrm{vw}} + s_{\mathrm{wu}}) - (s_{\mathrm{vk}} + s_{\mathrm{ku}})].$$

The decrease in switching capacitance of $R^{\mathrm{T}}_{\min}$ due to swapping w and k is $D^{v}_{\min}$, where

$$D^{v}_{\min} = [(s_{\mathrm{jk}} + s_{\mathrm{ki}}) - (s_{\mathrm{jw}} + s_{\mathrm{wi}})].$$

Now, let us assume that an operation v from $R^{\mathrm{T}}_{\max}$ conflicts with operation j on $R^{\mathrm{T}}_{\min}$. In this case, we try to reschedule operations v or j. Let operation v be rescheduled such that its new start time is $T'_s(v)$ and finish time is $T'_f(v)$. Let the immediate predecessor operations of v be denoted as $\pi(v)$ and immediate successor operations of v be denoted as $\mu(v)$. The reschedule is feasible only if

$$1) \qquad T_f(\pi(v)) \leq T'_s(v)$$

and

$$2) \qquad T_s(\mu(v)) \geq T'_f(v).$$

However, we only need to check condition 1) if $T'_s(v) < T_s(v)$ and check condition 2) if $T'_s(v) > T_s(v)$. Further, all the rescheduling that will be attempted are soft, i.e., the operation v is not actually rescheduled unless v is actually selected for a move and rescheduling is necessary for an *insert*.

First, we try to find the idle period (equal to the execution time of v) on resource $R^{\mathrm{T}}_{\min}$ closest to $T_s(v)$ such that v can be rescheduled and executed on $R^{\mathrm{T}}_{\min}$. There can be two cases.

1) The new start time $T'_s(v) < T_s(j)$ as shown in Fig. 4(b), then $I^{v\prime}_{\min} = [(s_{\mathrm{iv}} + s_{\mathrm{vj}}) - s_{\mathrm{ij}}]$.
2) The start time $T'_s(v) > T_s(j)$ as shown in Fig. 4(c), then $I^{v\prime\prime}_{\min} = [(s_{\mathrm{jv}} + s_{\mathrm{vk}}) - s_{\mathrm{jk}}]$.

If both rescheduling cases are feasible, we evaluate both $I^{v\prime}_{\min}$ and $I^{v\prime\prime}_{\min}$ and choose the soft-reschedule for v that yields the minimum $I^{v}_{\min}$.

If rescheduling is not feasible for v, then we try to reschedule conflicting operation j or conflict (v) at the destination. First, we try to find the closest idle period on $R^{\mathrm{T}}_{\min}$ such that j can

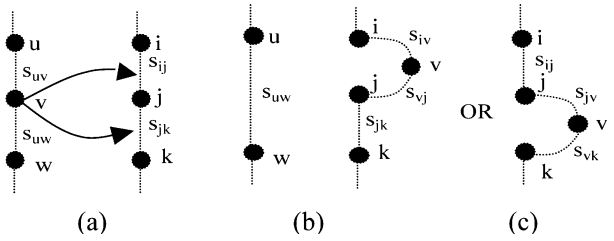

Fig. 4. Conflict insert: rescheduling of the incoming operation.

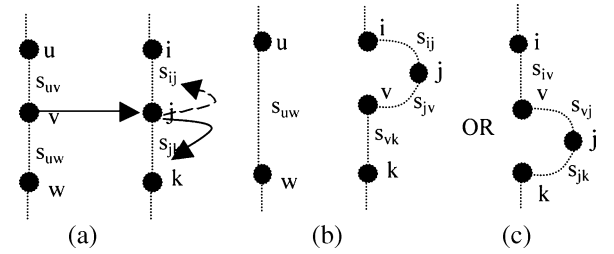

Fig. 5. Conflict insert: rescheduling of the operation on the destination functional unit.

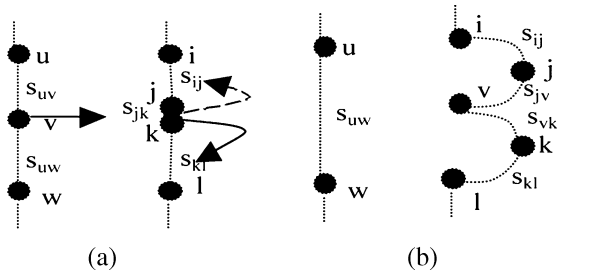

Fig. 6. Conflict insert: rescheduling of the operations on the destination functional unit for multiple conflicts.

Input: Scheduled Data Flow Graph(DFG)
Estimated Switching capacitance between the operations

Begin:
For each operation type
Build comparability graph for the operation
Label the edges with the estimated switching capacitance
Generate flow based optimal low power binding
end For
Generate power trace file and floorplan using HotFloorplan
Perform temperature simulation using HotSpot
While(same type ($R^T_{max} - R^T_{min}$) > T_{diff} constraint)
For each operation j $\in R^T_{max}$
Initialize D^j_{max} to -∞ and I^j_{min} to +∞
Evaluate D^j_{max} for deleting j from R^T_{max}
If (conflict(j) exits on R^T_{min})
If swap allowed for j, mark j for swap and break
else Soft-Reschedule j for inserting in R^T_{min}
If (reschedule(j) failed)
Soft-Reschedule operations conflict(j) on R^T_{min}
else continue;
Evaluate I^j_{min} for inserting j in R^T_{min}
end For
Choose j s.t it is marked swap or
max($(D^j_{max} - I^j_{min})$) ∀ j $\in R^T_{max}$ and Insert j to R^T_{min}
Reschedule j or conflict(j) if there is a Soft-Reschedule
Generate power trace file and perform temperature simulation using HotSpot
Evaluate solution
If no improvement, backtrack to previous solution and break
end While
End

Output: Uniform temperature profile of functional units

Fig. 7. Outline of the thermal feedback driven rescheduling rebinding algorithm.

be rescheduled. Similarly as before, if $T'_s(j) < T_s(j)$ and is feasible, then $I^{v\prime}_{\min} = [(s_{\rm jv} + s_{\rm vk}) - s_{\rm jk}]$ as shown in Fig. 5(b). If $T'_s(j) > T_s(j)$, and feasible then $I^{v\prime\prime}_{\min} = [(s_{\rm iv} + s_{\rm vj}) - s_{\rm ij}]$ as shown in Fig. 5(c). If both moves are feasible, we choose the one yielding the minimum $I^v_{\min}$.

Lemma 1: For multicycle operations there can be at most two conflict operations onto $R^{\rm T}_{\min}$ for v.

This is true since we assume that there is a single architecture for a resource type. Thus, the execution period of operations on $R^{\rm T}_{\max}$ and $R^{\rm T}_{\min}$ is the same and there can be at most two operations j and k on $\rm RT_{min}$ such that $T_s(j) < T_s(v) < T_s(k) < T_f(v)$ or $T_s(v) < T_f(j) < T_s(k) < T_f(v)$.

Assume there is no feasible reschedule for v, and the operations j and k conflict with v such that $T_s(j) < T_s(k)$. We then try to find a feasible earlier start time for $j - T'_s(j) < {\rm Ts}(j)$ and a later start time for $k - T'_s(k) > T_s(k)$.

If $[T'_s(k) - T'_f(j)] \geq$ execution period of v, then $I^v_{\min} = [(s_{\rm jv} + s_{\rm vk}) - s_{\rm jk}]$. This is shown in Fig. 6(a) and (b). We refer to the inserts shown in Figs. 4–6 as *conflict inserts*.

Next, we explain the overall flow of our algorithm. The outline our algorithm is shown in Fig. 7. Starting from a switching optimized low-power binding using min-cost flow as proposed by Chang and Pedram [30], we first generate the temperature profile of the resources using thermal simulation. The thermal aware floorplan prior to thermal simulation is generated using HotFloorplan [29].

For each resource set, we select the resource with highest temperature $R^{\rm T}_{\max}$ and resource with the least temperature $R^{\rm T}_{\min}$. If the temperature difference between $R^{\rm T}_{\max}$ and $R^{\rm T}_{\min}$ is greater than the given constraint, then we start our optimization. For each operation $v \in R^{\rm T}_{\max}$, we calculate the decrease in switching capacitance $D^v_{\max}$ for deleting v. We also calculate the increase in switched capacitance $I^v_{\min}$ of $R^{\rm T}_{\min}$ for inserting v. The insert is a *compatible insert* if v is compatible with $R^{\rm T}_{\min}$. If there exits a conflict operation j, then we check whether *swap* is allowed, i.e., whether such a move decreases inter-iteration switching of $R^{\rm T}_{\min}$ and overall switching of $R^{\rm T}_{\max}$. If *swap* is found feasible, operations v and j are exchanged. If swap is not feasible, then it is a case for *conflict insert*. We calculate $I^v_{\min}$ for *conflict insert* of v. If both insert and swap fail for v, operation v cannot be reassigned to $R^{\rm T}_{\min}$. For all operations $v \in R^{\rm T}_{\max}$ for which insert is allowed, we calculate $D^v_{\max}$ and $I^v_{\min}$. Then we chose v^* for which $(D^v_{\max} - I^v_{\min})$ is maximum for the move. If it is a *compatible insert* then no rescheduling is necessary. On the other hand, if it is a *conflict insert* then the soft rescheduling information associated with v^* is used to actually reschedule either v^*, or conflict operation(s) of v^* on $R^{\rm T}_{\min}$. Thus, we have a new binding solution for each resource type.

We then perform thermal simulation and obtain the temperature profile of the resources and try to reschedule-rebind operations from the new $R^{\mathrm{T}}_{\max}$ to $R^{\mathrm{T}}_{\min}$. Thus, based on thermal feedback, we iteratively minimize the temperature of the hottest resource. If in a certain step the solution quality degrades compared to the previous step, we backtrack to the previous solution and end our iteration. Otherwise, we iterate until the difference between $R^{\mathrm{T}}_{\max}$ and $R^{\mathrm{T}}_{\min}$ is less than the given constraint or a fixed number of iterations have been performed.

Initial low-power binding by min-cost flow technique is solvable in polynomial time. Each iteration of optimization examines rebinding (and rescheduling), which is bound by $O(V)$ time (where V is the number of operations). In addition, there is computation dictated by thermal simulation in each iteration and initial thermal-aware floorplaning. The initial thermal-aware floorplan remains unchanged in our flow. There can be more opportunities for thermal management in the synthesis flow if thermal-aware floorplanning is performed at each of the intermediate steps. However, this has significant runtime implications. Design space exploration is performed during the architectural synthesis phase. It is imperative that architectural synthesis is fast. Existing thermal-aware floorplanners based on simulated annealing are computationally expensive. Thus, we choose to perform thermal-aware floorplanning at the beginning of our thermal aware synthesis flow and the floorplan remains unchanged during the intermediate optimization stages.

In our approach, there are two major improvements over temperature reduction by iterative task reassignment [27], which is equivalent to load balancing with respect to switching activity. First, instead of the switching power, the temperature of the resource (from thermal simulation) is used for task rebinding. The temperature of a resource depends on power density, geometry, and its relative location in the floorplan. Thus, thermal simulation accurately identifies the hottest resource. Second, during rebinding, we allow rescheduling in case of conflicts. We will show the improvement in peak temperature reduction using our integrated thermal management scheme as compared to the load balancing approach without thermal simulation in the results section.

IV. Experimental Results

In the following sections, we will first describe our experimental flow and then we will present our results.

A. Experimental Flow

Our overall flow is shown in Fig. 8. Our experiments were performed on applications from the MediaBench suite [31]. We extracted the DFGs of representative functions from MediaBench applications using SUIF and Machine-SUIF compiler infrastructure [32]. We also performed our experiments on large synthetic DFGs generated using task graphs for free (TGFF) [33]. TGFF generates pseudorandom task graphs (DAGs) in accordance with the user's parameterized graph and database specifications.

The DFGs were then scheduled using the resource constrained latency minimizing scheduler. Next, we estimated switching activities between pairs of operations that can potentially be executed on the same resource consecutively. We simulated the input DFGs to generate switching probabilities for individual operations using a trace of 10 000 input values. For MediaBench suite our resource set included multipliers and ALUs. The operations of DFGs generated using TGFF were mapped onto ALUs. ALUs execute arithmetic operations. Multipliers and ALUs were synthesized using Synopsys Design Compiler onto a 180-nm technology library and capacitance values of the functional modules have been extracted. Scaling trends presented in [34] were used to get switched capacitance and nominal power values for a switching activity of 0.5 at 130-nm technology node for each resource type. The size of the synthesized ALU was 2 mm^2 and the dynamic power consumption varies between 1.2 and 3.7 W depending on the type and switching of the operations bound on the resource. The scaled capacitance was combined with bit toggle probabilities obtained through simulation to generate switched capacitance values at a 130-nm technology node.

We then created a comparability graph for the scheduled DFG for each operation type. The edges in the comparability graph are labeled with the estimated switched capacitance between two consecutive operations. The low power binding is based

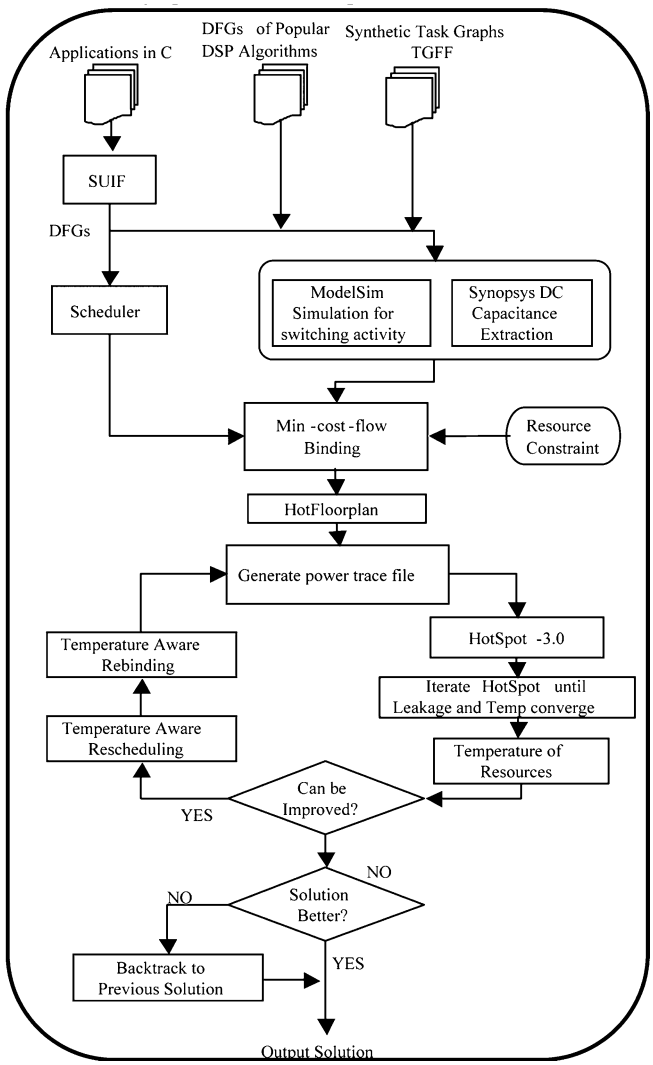

Fig. 8. Experimental flow.

on this comparability graph. We first generate binding solutions using flow formulation as proposed by Chang and Pedram [30]. We solved the network flow formulation using a software package developed by Goldberg [35].

After binding, we use HotFloorplan [29] to create a thermal-aware floorplan. HotFloorplan manages the lateral heat propagation during the floorplanning stage. It extends the simulated annealing algorithm for slicing floorplans and incorporates temperature as a cost function using HotSpot thermal simulator. As an input to the floorplanner, we generate the average switching power of the resources and connectivity information between the functional units. We have used HotFloorplan to create an initial thermal-aware floorplan, which remains unchanged in the successive optimization steps. Simulated annealing and thermal simulation is computationally expensive. For example, HotFloorplan for an Alpha 21364 processor with 16 processor components typically takes about 45 min to complete on a 2-GHz Athlon. It is imperative that high-level synthesis is fast to allow extensive design space exploration. For the task graphs, our resource set had 20 to 40 functional units and our experience with the runtimes for HotFloorplan has been similar (in the order of hours). Therefore, we perform thermal aware floorplanning once at the beginning of our optimization flow (with fast cooling for simulated annealing).

We use HotSpot [18] to measure the temperatures of functional units. HotSpot is originally developed to model the temperature of a microprocessor at the granularity of functional units by making use of the duality that exists between heat flow and electricity. It constructs an *RC* network of thermal resistances and capacitances of the functional units and uses circuit-solving techniques to obtain the temperatures at the centers of the functional units. It takes as input, the floorplan and the initial temperatures of the functional units, the heat spreader, and the heat sink. The power values of each functional unit are input in the form of a trace file where each line corresponds to a sampling interval. We have used a HotFloorplan to generate thermal-aware floorplan for our designs. We have assumed the same chip-packaging configuration as modeled by HotSpot. The initial temperatures of each functional unit are set at 77 °C and the heat spreader and heat sink at 47 °C. The values of some of these parameters (such as chip area, sink, and spreader size, etc.) have been scaled to appropriate values for our synthesized designs. Finally, HotSpot simulates the activity on the chip and computes the steady-state temperatures of the functional units.

The default chip packing configuration in HotSpot represents that of the Alpha 21364 processor. We have assumed the same chip packaging configuration as a representative of the embedded system onto which the design will be synthesized. The chip is packaged with the die placed against a spreader plate, often made of aluminum, copper, or some other highly conductive material, which is in turn placed against a heat sink of aluminum or copper that is cooled by a fan. A typical example is shown in Fig. 9.

A single thermal resistance $R_{\text{convection}}$ represents the convective heat transfer from the package to the air. Air is assumed to be at a fixed ambient temperature, which is often assumed in thermal design to be 45 °C. Different HotSpot parameters are

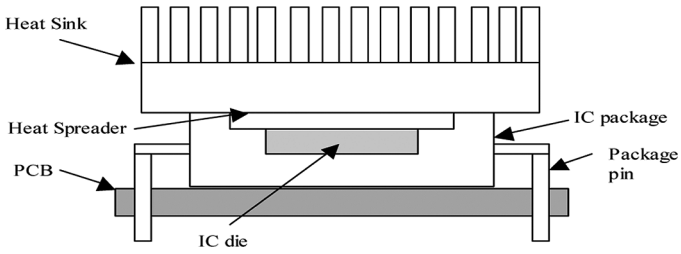

Fig. 9. Side view of a typical chip package.

$C_{convection}$ = 140.4 J/K	$R_{convection}$ = 0.2 K/W
Heat sink side = 30 mm	Heat sink thickness = 6.9 mm
Spreader side = 20 mm	Spreader thickness = 1 mm
Chip thickness = 0.5mm	Sampling interval = 3.33 us

Fig. 10. HotSpot parameters.

shown in Fig. 10. The values of some of these parameters (such as chip area, sink, and spreader size, etc.) have been scaled to appropriate values for our synthesized designs.

Based on the feedback on the temperature of the resources, our thermal management scheme reschedules and rebinds operations from the highest temperature resource onto the least temperature resource, evaluating the "goodness" of the new solution at each step by thermal simulation. The algorithm finally generates a solution, which effectively minimizes the temperature of the hottest resource.

We generated the temperature profile and computed the power of the resources using two approaches. First, we generate one solution using our TA-HLS technique. Then, we used flow-based low-power binding [30] within TU-HLS to generate another solution. We evaluated the effectiveness of our algorithm by comparing the results from TA-HLS with TU-HLS.

B. Results

In this section, we discuss the experimental results. First, we performed experiments with relatively small DFGs from MediaBench benchmark suite [31]. Without thermal-aware floorplaning, integrated thermal management technique performed better than previously proposed solutions [26], [27]. The maximum peak temperature reduction across benchmarks for integrated thermal management technique was 26.6 °C as compared to 19.86 °C for the iterative binding approach [27]. The maximum peak temperature reduction obtained in [26] was 11.9 °C for temperature constrained resource minimization allocation and binding. The temperature was calculated using analytical models, which failed to capture lateral heat spreading and geometry of the functional units.

For these small benchmarks, thermal-aware floorplanning mostly alleviated hotspot formation and our integrated thermal management approach did not show much improvement. Next, we applied our integrated thermal management on large DFGs generated using TGFF [33]. We have performed experiments using DFGs having 100 to 200 nodes. The operations were bound to ALUs. Table I shows the number of ALUs used by

TABLE I
RESOURCE REQUIREMENTS FOR BENCHMARKS

	Tgff_100	Tgff_110	Tgff_130	Tgff_140	Tgff_160	Tgff_170	Tgff_180	Tgff_190	Tgff_200
ALU	20	22	26	28	32	34	36	38	40

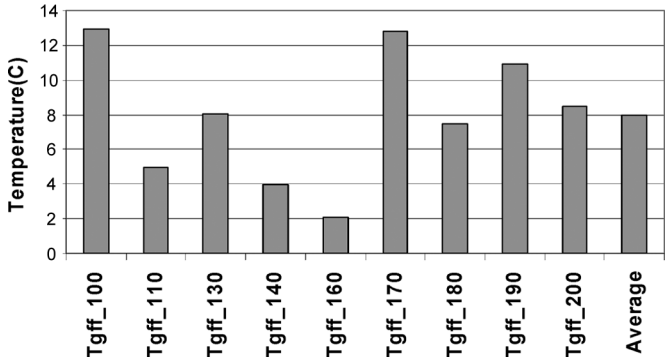

Fig. 11. Temperature reduction for the hottest resource across benchmarks.

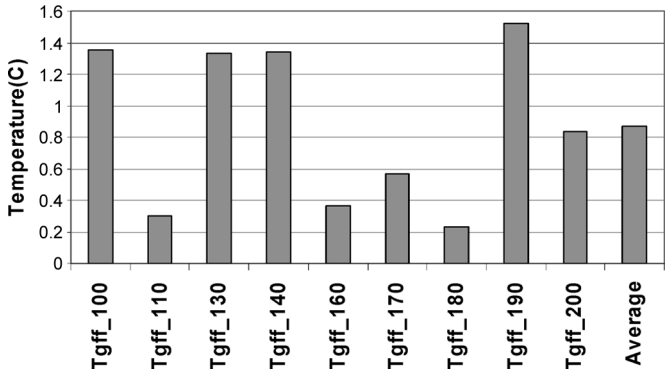

Fig. 12. Difference between average temperature of all resources from TA-HLS and TU-HLS.

our benchmarks. These set of benchmarks show the most interesting set of results. We demonstrate with the following results that our integrated thermal management approach achieves an effective control of peak temperature and creates even temperature profile of the resources. We improve the quality further even in the presence of a good initial solution provided by thermal-aware floorplanning.

1) Temperature Reduction: Fig. 11 shows the temperature reduction for the hottest resource across the benchmarks by using TA-HLS in comparison to TU-HLS. The maximum reduction obtained is 12.94 °C for Tgff_100 and the average temperature reduction is 7.95 °C. Our approach was able to effectively decrease the temperature of the hottest resource across benchmarks.

In Fig. 12, we show the average temperature increase of the resources across benchmarks using TA-HLS flow versus TU-HLS flow. The mean of average temperature increase is only 0.87 °C. When we combine our results presented in Fig. 11 (reduction in temperature of the hottest resource) and the fact that the average temperature across all resources increases only by a very small amount, we reach the following conclusion.

TABLE II
PERCENTAGE IMPROVEMENT IN LEAKAGE POWER AT 130- AND 100-nm TECHNOLOGY NODES

Benchmarks	% Change in Switching	% Leakage Imp at 130 nm	% Leakage Imp at 100 nm
Tgff_100	-4.11	5.04	6.04
Tgff_110	-0.83	2.83	2.99
Tgff_130	-3.01	1.69	3.47
Tgff_140	-2.12	0.75	1.33
Tgff_160	-0.53	1.51	1.82
Tgff_170	-1.65	7.16	9.62
Tgff_180	-0.34	6.36	8.10
Tgff_190	-2.46	1.81	3.49
Tgff_200	-1.52	4.27	5.54
Average	**-1.84**	**3.49**	**4.71**

The probability of hotspot occurrence in designs synthesized through TA-HLS is reduced significantly. Next, we present the impact of thermal management scheme on power consumption.

2) Power Minimization by Leakage Control: Table II shows the impact of our TA-HLS flow on power compared to TU-HLS flow. First, we discuss the impact on dynamic power. Negative values for dynamic power indicate that the switching power has increased in TA-HLS. This is expected since we are considering a switching optimized low-power binding as TU-HLS. Our iterative rebinding rescheduling decisions might degrade the optimal switching solution in the process. Our results show that the dynamic power, on average, increases by 1.84% for TA-HLS. Next, we considered the impact of out techniques on leakage power. We anticipate some savings due to the decrease in peak temperatures. On the other hand, the desired smooth thermal profile is created by allowing the temperature of relatively cooler resources to increase in some cases. Our results demonstrate that overall, our technique does not incur any overhead in power; in fact it helps decrease total leakage power to some extent. Note that, our primary aim was to maintain control over peak temperatures and positive impact on leakage is a byproduct of this effort in this case. Note that since our framework operates under fixed resource constraints, this is achieved virtually free.

We have used scaling trends presented in [34] and [36] to obtain the relation between temperature and leakage power at 130- and 100-nm technology nodes. At 130 nm we have, on average, a 3.49% reduction in leakage power using TA-HLS. At 100 nm we achieve on average a 4.71% reduction in leakage power. Fig. 13 shows the percentage improvement in total power at 100-nm technology node using TA-HLS when compared to TU-HLS. It shows that the maximum gain in total power at 100 nm can be as high as 5.94% using our integrated thermal management scheme. We observe that at smaller technology nodes the reduction in leakage power will increase further.

V. CONCLUSIONS

In this paper, we have presented an integrated thermal management in high-level synthesis. Our main goal is to reduce the peak temperature of the functional units and obtain an even thermal profile of the resources. We have shown that our TA-HLS flow is indeed effective in preventing hotspot formation with associated savings in leakage power.

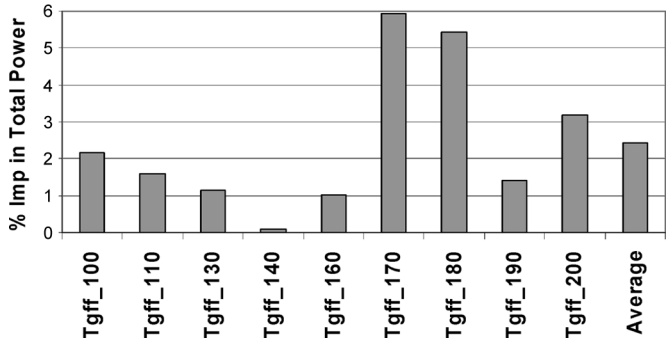

Fig. 13. Percentage improvement in total power using TA-HLS compared to TU-HLS at 100-nm generation.

We have compared TA-HLS with a TU-HLS. We show a reduction in the peak temperature of functional units by as high as 12.94 °C and on average by 7.95 °C and obtain an even temperature profile of the resources. Using TA-HLS, there is an average 4.71% reduction in leakage at 100-nm technology node over a switching optimized HLS flow. The impact of our technique on the overall power consumption is virtually overhead free. Similarly, we achieve the control over peak temperatures without using additional resources.

ACKNOWLEDGMENT

The authors would like to thank the reviewers of this paper and Dr. G. Memik for their helpful comments and suggestions.

REFERENCES

[1] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. Int. Symp. Comput. Arch.*, 2003, pp. 2–13.

[2] S. Gunther, F. Binns, D. M. Carmean, and J. C. Hall, "Managing the impact of increasing microprocessor power consumption," *Intel Technol. J.*, vol. 5, no. 1, pp. 1–9, Feb. 2001.

[3] M. Xu and F. J. Kurdahi, "Layout-driven RTL binding techniques for high-level synthesis using accurate estimators," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 2, no. 4, pp. 312–343, 1997.

[4] L. Zhong and N. K. Zha, "Interconnect-aware high-level synthesis for low power," in *Proc. Int. Conf. Comput.-Aided Des.*, 2002, pp. 110–117.

[5] C. H. Gebotys and M. I. Elmasry, "Simultaneous scheduling and allocation for cost constrained optimal architectural synthesis," in *Proc. Des. Autom. Conf.*, 1991, pp. 2–7.

[6] A. Dasgupta and R. Karri, "Simultaneous scheduling and binding for power minimization during microarchitecture synthesis," in *Proc. Int. Symp. Low Power Electron. Des.*, 1995, pp. 69–74.

[7] J. P. Weng and A. C. Parker, "3D scheduling: high-level synthesis with floorplanning," in *Proc. Des. Autom. Conf.*, 1991, pp. 668–673.

[8] P. Prabhakaran and P. Banerjee, "Simultaneous scheduling, binding and floorplanning in high-level synthesis," in *Proc. Int. Conf. VLSI Des.*, 1998, pp. 428–434.

[9] S. Hong and T. Kim, "Bus optimization for low-power data path synthesis based on network flow method," in *Proc. Int. Conf. Comput.-Aided Des.*, 2000, pp. 312–317.

[10] C. G. Lyuh, T. Kim, and C. L. Liu, "An integrated data path optimization for low power based on network flow method," in *Proc. Int. Conf. Comput.-Aided Des.*, 2001, pp. 553–559.

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

[12] J. Cong and Y. Zhang, "Thermal-driven multilevel routing for 3-D ICs," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2005, pp. 121–126.

[13] C. Tsai and S. Kang, "Standard cell placement for even on-chip thermal distribution," in *Proc. Int. Symp. Phys. Des.*, 1999, pp. 179–184.

[14] C. C. N. Chu and D. F. Wong, "A matrix synthesis approach to thermal placement," in *Proc. Int. Symp. Phys. Des.*, 1997, pp. 163–168.

[15] A. Basu, S. Lin, V. Wason, A. Mehrotra, and K. Banerjee, "Simultaneous optimization of supply and threshold voltages for low-power and high-performance circuits in the leakage dominant era," in *Proc. Des. Autom. Conf.*, 2004, pp. 884–887.

[16] K. Banerjee, S.-C. Lin, and V. Wason, "Leakage and variation aware thermal management of nanometer scale ICs," in *Proc. IMAPS-Adv. Technol. Workshop Thermal Managem.*, 2004, pp. 1–5.

[17] M. N. Sabry, "Dynamic compact thermal models: An overview of current and potential advances," in *Proc. Int. Workshop Thermal Investigations ICs Syst.*, 2002, pp. 100–104.

[18] W. Huang, S. Ghosh, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Compact thermal modeling for temperature-aware design," in *Proc. Des. Autom. Conf.*, 2004, pp. 878–883.

[19] W. Liao, F. Lei, and L. He, "Microarchitecture level power and thermal simulation considering temperature dependent leakage model," in *Proc. Int. Symp. Low Power Electron. Des.*, 2003, pp. 211–216.

[20] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proc. Int. Symp. High-Performance Comput. Arch.*, 2001, pp. 171–182.

[21] L. Cao, J. Krusius, M. Korhonen, and T. Fisher, "Transient thermal management of portable electronics using heat storage and dynamic power dissipation control," *IEEE Trans. Compon., Packag., Manuf. Technol. A*, vol. 21, no. 1, pp. 113–123, Jan. 1998.

[22] W. Huang, J. Renau, S.-M. Yoo, and J. Torellas, "A framework for dynamic energy efficiency and temperature management," in *Proc. Int. Symp. Microarch.*, 2000, pp. 202–213.

[23] L. Shang, L. S. Peh, A. Kumar, and N. K. Jha, "Thermal modeling, characterization and management of on-chip networks," in *Proc. Int. Symp. Microarch.*, 2004, pp. 67–78.

[24] Y. Yang, Z. Gu, C. Zhu, L. Shang, and R. P. Dick, "Adaptive chip-package thermal analysis for synthesis and design," in *Proc. Des. Autom. Test Eur.*, 2006, pp. 844–849.

[25] W.-L. Hung, Y. Xie, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "Thermal-aware task allocation and scheduling for embedded systems," in *Proc. Des. Autom. Test Eur.*, 2005, pp. 898–899.

[26] R. Mukherjee, S. O. Memik, and G. Memik, "Temperature-aware resource allocation and binding in high-level synthesis," in *Proc. Des. Autom. Conf.*, 2005, pp. 196–201.

[27] ——, "Peak temperature control and leakage reduction during binding in high level synthesis," in *Proc. Int. Symp. Low Power Electron. Des.*, 2005, pp. 251–256.

[28] Z. P. Gu, Y. Yang, J. Wang, R. P. Dick, and L. Shang, "TAPHS: Thermal-aware unified physical-level and high-level synthesis," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2006, pp. 879–885.

[29] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *J. Instruction-Level Parallelism*, vol. 7, pp. 1–16, 2005.

[30] J. M. Chang and M. Pedram, "Register allocation and binding for low power," in *Proc. Des. Autom. Conf.*, 1995, pp. 29–35.

[31] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: A tool for evaluating and synthesizing multimedia and communicatons systems," in *Proc. Int. Symp. Microarch.*, 1997, pp. 330–335.

[32] Stanford Univ. Compiler Group, Palo Alto, CA, The SUIF 2 Compiler System (1999). [Online]. Available: http://suif.stanford.edu/suif/suif2/index.html

[33] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. Int. Workshop Hardw./Softw. Codes.*, 1998, pp. 97–101.

[34] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, Jul./Aug. 1999.

[35] A. V. Goldberg, "An efficient implementation of a scaling minimum-cost flow algorithm," *J. Algorithms*, vol. 22, no. 1, pp. 1–29, Jan. 1997.

[36] F. Fallah and M. Pedram, "Standby and active leakage current control and minimization in CMOS VLSI circuits," *Special Low-Power LSI Issue: IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, pp. 509–519, 2005.

Rajarshi Mukherjee (M'01) received the B.E.Tel.E. degree in electronics and telecommunication engineering (Hons.) from Jadavpur University, Calcutta, India, in 2000, and the M.S. and Ph.D. degrees from Northwestern University, Evanston, IL, in 2003 and 2006, respectively.

He is presently a Senior Research and Development Engineer at Synopsys Inc., Mountain View, CA. Before joining the graduate program at Northwestern University, he was with Texas Instruments Incorporated, Bangalore, India, from 2000 to 2001. His research interests include thermal-aware design and analysis techniques for integrated circuits and high-performance microprocessors, timing analysis, and high-level synthesis.

Dr. Mukherjee was a recipient of the Walter Murphy Fellowship from Northwestern University in 2001–2002.

Seda Ogrenci Memik (SM'05) received the B.S. degree in electrical and electronic engineering from Bogazici University, Istanbul, Turkey, and the Ph.D. degree in computer science from University of California, Los Angeles.

She is an Assistant Professor at the Electrical Engineering and Computer Science Department of Northwestern University, Evanston, IL. Her research interests include embedded and reconfigurable computing, thermal-aware design automation, and thermal management for high performance microprocessor systems. She has served as technical program committee member, organizing committee member, and subcommittee chair of several conferences, including ICCAD, DATE, FPL, GLSVLSI, and ARC.

Dr. Memik was the recipient of the National Science Foundation Early Career Development (CAREER) Award in 2006.