Temperature-Aware Resource Allocation and Binding in High-Level Synthesis

Rajarshi Mukherjee, Seda Ogrenci Memik, and Gokhan Memik
Department of Electrical and Computer Engineering, Northwestern University, IL, USA
{rajarshi, seda, memik}@ece.northwestern.edu

ABSTRACT

Physical phenomena such as temperature have an increasingly important role in performance and reliability of modern process technologies. This trend will only strengthen with future generations. Attempts to minimize the design effort required for reaching closure in reliability and performance constraints are agreeing on the fact that higher levels of design abstractions need to be made aware of lower level physical phenomena. In this paper, we investigated techniques to incorporate temperature-awareness into high-level synthesis. Specifically, we developed two temperature-aware resource allocation and binding algorithms that aim to minimize the maximum temperature that can be reached by a resource in a design. Such a control scheme will have an impact on the prevention of hot spots, which in turn is one of the major hurdles in front of reliability for future integrated circuits. Our algorithms are able to reduce the maximum attained temperature by any module in a design by up to 19.6^{o}C compared to a binding that optimizes switching power.

Categories and Subject Descriptors: B.6.3 [**Hardware**]: Logic Design - Design Aids; J.6 [**Computer Applications**]: Computer-aided Engineering (CAD).

General Terms: Algorithms, Design, Experimentation.

Keywords: Binding, Temperature, Switching, Leakage.

1. INTRODUCTION

Transistor counts in all electronic systems are increasing steadily. The latest Intel Itanium 2 processor contains more than 200 Million transistors [12]. It is predicted that Moore's law will not slow down for at least another decade. By that time, integrated circuits (ICs) are expected to have feature sizes of 30 nanometers, allowing for integration of billions of devices on a single die and enabling unforeseen computational capabilities. One consequence of this trend is that heat dissipation on modern and future technologies is skyrocketing. ICs achieve extremely high computational power per unit silicon area at the expense of increased power densities. This in turn brings about the problem of potentially large operating temperature variations.

Temperature has a significant impact on circuit performance. Increase in temperature has an adverse effect on carrier mobility, hence, switching speed of transistors. Interconnect resistance increases with temperature as well. Circuit reliability is also heavily impacted by temperature. Regions on a chip that generate excessive amounts of heat are referred to as hot spots. Hot spots can jeopardize correct execution by causing transient as well as permanent faults. Even if excessive heat does not lead to spontaneous damage, it accelerates electromigration, which can lead to permanent damage in the long run.

Several aspects of concern for design of future ICs, from reliability to performance and manufacturing cost, call for effective control of temperature during the design process of ICs. Moreover, every effort must be made to incorporate awareness of this phenomenon into every level of design abstraction. Design paradigms that target performance centric metrics such as delay and power have already evolved to a point that closure in these metrics inevitably involves planning and optimization above and beyond the physical synthesis stage. Similarly, management of emerging physical issues such as temperature need to be taken into account at all levels of abstraction of the design process.

On a related track, power optimization and management techniques have long been studied for various design levels including high-level synthesis. However, power optimization does not specifically address the problem of regulating local activity in a design, which can lead to hot spots over longer periods of activity, while the total power consumption might be seemingly well-bounded.

In this paper, we present techniques to incorporate temperature awareness into high-level synthesis. Decisions made during high-level synthesis have great impact on the activity of functional resources. If tasks such as scheduling, resource allocation and binding are performed with such awareness, temperature increase can be controlled more effectively, especially locally, which is closely related to hot spot formation. Our specific contributions in this paper are as follows. We

- Formulate the temperature-aware resource binding problem,
- Develop models for thermal profiles of functional resources for a given task assignment,
- Develop resource binding techniques to effectively control maximum temperature of a binding
- Study the impact of temperature on leakage current. Specifically we compare the performance of a switching optimized binding and our temperature-aware binding with respect to total power consumption, where the leakage power has a growing contribution with increasing temperature.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work. In Section 3 we elaborate on the temperature-aware design paradigm in high-level synthesis. In Section 3.1 we discuss some relevant assumptions in our work and introduce the temperature model we have used. We introduce our

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

temperature-aware allocation and binding algorithm in Section 3.2. Section 4 presents our experimental flow and results. We conclude our paper with a summary in Section 5.

2. RELATED WORK

There are various techniques to design packages and cooling techniques to develop better heat removal capability. A large amount of research effort has been spent on development of circuit and board level models of heating and thermal distribution. For design and analysis of high performance microprocessors various techniques to model thermal effects have been developed [18], [11], [15]. Runtime thermal management via clock gating using real-time temperature sensing has been included in Intel Pentium 4 processors [9]. Other techniques such as frequency and/or voltage scaling, sub-banking, etc. have been investigated [2], [10], [3]. Skadron et al. developed a thermal model for microprocessors to capture hotspots and aid thermal management [20].

From the other end of design hierarchy, physical design tools have been proposed to enable even thermal distribution on chips. Tsai and Kang developed a standard cell placement tool for even on-chip thermal distribution [22]. Chu and Wong proposed a matrix synthesis approach to thermal placement [6]. Cong et al. introduced a thermal-driven floorplanning algorithm for 3-D ICs [7]. Basu et al. introduced the electrothermal energy-delay-product optimization scheme to perform simultaneous optimization of supply and threshold voltages in CMOS circuits [1].

To the best of our knowledge so far it has not been attempted to incorporate thermal considerations into higher level synthesis tasks. In this work we aim to accomplish this goal by developing temperature aware resource allocation and binding algorithms to be deployed within high-level synthesis.

3. TEMPERATURE-AWARE HIGH-LEVEL SYNTHESIS

During high-level synthesis, decisions regarding distribution of tasks across functional modules as well as relative timing of individual tasks are made. Activity of modules throughout the execution of an application is thereby determined. Intuitively, the higher the number of tasks assigned to a module the more activity will be observed in that module. In the context of temperature, we associate switching power dissipation during execution of a task with activity of the module. This in turn causes the module to generate heat and consequently the temperature is likely to increase. Also, the relative timing of operations executing on a given module is important. Consider two scenarios, where first, a module executes a set of tasks in a certain interval of time. In the second case, the same module executes the operations across a period of time, where there are more inactive time intervals in between operations. The rise in temperature in the first case would be expected to be steeper, since the module has less opportunity to relieve any heat buildup during execution. On the other hand, a careful assignment of tasks to the module considering the accumulative effects of continuous activity on heat buildup could help control the temperature rise.

Since temperature is related to switching activity of resources, one could argue that minimizing total switching activity should help bound the temperature rise. This is indeed the case: minimizing switching activity also helps keep the temperature low. However, as we will show in the following sections a temperature-aware binding controls the maximum temperature reached in a design more effectively. Thereby it minimizes the likelihood of occurrence of hot spots, which is not considered by switching minimization schemes. Unlike switching power, temperature has an additive nature, i.e., the temperature at a given point in time will depend on the entire history of activity in the past. If this is not taken into account, cumulative heat will cause hot spots although the average activity seems to be well bounded.

In the following sections we will present our temperature-aware resource allocation and binding scheme. First, we will describe our underlying assumptions and discuss our temperature model. Next, we will present our temperature-aware binding algorithm.

3.1 Temperature Model and Relevant Assumptions

In this work, we focus on the resource allocation and binding steps and hence, assume that scheduling has already been done. We do not make changes at the scheduling stage. Both scheduling and binding are hard tasks individually, often times aiming to optimize multiple objectives simultaneously. While it is part of our long-term agenda to incorporate temperature-awareness into other stages of high-level synthesis also, we determined it to be reasonable to separate these two hard tasks initially and tackle each individually. Therefore, we directed our attention to resource binding first, which highly impacts the activity on a resource. In addition, there are well-established network flow-based techniques to minimize switching activity during resource binding [5], [16], which relates to temperature as we discussed earlier. By developing an alternative temperature-aware resource binding scheme we were able to compare our approach to a binding that aims to minimize switching activity, where the starting point to both our approach as well as the minimum switching binding would be the same initial schedule. We will describe the network flow based binding technique in more detail in Section 4.1.

To evaluate the effect of binding on temperature we used a temperature model that represents increase in module temperature due to power dissipated during switching activity and decrease in temperature during idle cycles through conduction. Our model is rooted at the heat transfer theory, which establishes the duality between heat transfer and behavior of RC circuits [13]. Similar models have been used in the past for thermal modeling of microarchitectures [15], [11]. Variation in temperature can be modeled using an exponential transient behavior governed by a time constant analogous to the electrical time constant RC. This time constant will determine the speed at which heat will cause an increase in temperature. Similarly cooling can be modeled with a transient behavior. We refer to the components of the RC time constant as the thermal resistance R and the thermal capacitance C of a module. We evaluate the change in temperature at the granularity of one operation executed by the resource. Also, in our model the temperature variations of individual modules are evaluated independently. Then, the temperature variation that occurs due to the resource executing an operation takes the following form:

$$T_{i+1} = T_A + (T_i - T_A)e^{-\frac{d_{i,i+1}}{RC}} + \Delta T_{tot}$$

T_A is the ambient temperature, T_i is the temperature of a module after having executed operation i, T_{i+1} is the temperature of the module after executing operation $i+1$ following operation i, $d_{i,\,i+1}$ is

the difference between the finish time of operation *i* and the start time of operation *i+1*, and *R* and *C* are the thermal resistance and capacitance of the resource respectively. The values of R and C have been derived using a similar method used by Huang et al. [11]. Thermal resistance is proportional to the chip thickness and inversely proportional to the cross-sectional area of the resource. Thermal capacitance is proportional to the chip thickness and cross-sectional area. We have calculated the RC constant to be approximately 232μs for silicon of 0.5mm die thickness. ΔT_{tot} is the temperature contribution due to the total power dissipation of the resource. We calculate ΔT_{tot} using the formula used in [19] as shown below

$$\Delta T_{tot} = \frac{P_{tot}\Delta t}{C} + \frac{T_i \Delta t}{R\,C}$$

where $P_{tot} = P_{leakage} + P_{dynamic}$ is the total power and Δt is one clock cycle duration. However the temperature rise of an IC is similar to capacitance charging, the temperature getting saturated after having reached a maximum operating temperature that we assume to be 400K (127 ^{o}C). We use first order approximation to calculate the actual contribution of ΔT_{tot}.

Leakage power model depends exponentially on threshold voltage V_{th} and temperature T. We choose the following model template

$$P_{leakage} = \exp(f(V_{th}, T))$$

We have used a 4th order polynomial to represent $f(V_{th}, T)$. For our process technology at 180nm, this roughly corresponds to 15% of the dynamic power at the ambient temperature and doubles every 25^{o}C.

In the next section we will present our temperature-aware resource allocation and binding technique.

3.2 Temperature-Aware Resource Allocation and Binding

Our algorithm takes in a scheduled data flow graph (DFG) and the switching activity between the operations. It can operate under one of the following two alternative modes:

- Temperature_Constrained_Resource_Minimization: This mode tries to find a binding that does not allow the temperature of any resource to reach above a given threshold value. This constraint might force the binding algorithm to increase the number of resources and our algorithm tries to keep this increase minimum
- Resource_Constrained_Temperature_Minimization: In the second mode, our algorithm conforms to a given resource constraint while minimizing the maximum temperature reached by any resource.

For either mode of optimization, we perform a preprocessing of the scheduled DFG first. For each resource type we create a comparability graph of the operations that this resource can execute. A comparability graph is essentially a compatibility graph with transitive orientation. If operations *u* and *v* are compatible, there exists a directed edge from *u* to *v* if

start time (v) > finish time (u)

Each edge of the comparability graph has a weight equal to the switching activity if the two operations are bound to the same resource consecutively. The operations bound to a resource will be executed in the topological order dictated by the comparability graph. The comparability graph is essentially a directed acyclic graph (DAG). We first perform topological sort on this graph. Next, we visit the vertices of the comparability graph in topological order and we determine a parent for every vertex, which indicates that if an operation were assigned to a particular resource, that parent would be the best candidate to have been assigned on the same resource prior to this operation. We call the determination of a parent for each vertex *relaxation*. The relaxation idea is same as Dijkstra's shortest path algorithm where for each vertex the best parent is determined through which we could reach the vertex with the shortest distance from a start point. In the context of our problem we relax vertices with a different criterion. Say vertices $v_{i...j}$ can be reached from u. Temperature of the resource $(T_{i+1})_{i...j}$ is computed over a period of time to check the rise of the temperature of the module for executing (u, v_i), …(u, v_j) consecutively. For resource constrained temperature minimization, we relax those vertices in $v_{i...j}$ which lead to minimum rise of temperature. For temperature constrained resource minimization, we relax only those vertices that do not violate the temperature constraint. Whenever a vertex v_i is relaxed, the path length of the vertex is updated at $d[v_i] = d[u] + 1$ and the parent of v_i is set to u. The relaxation pseudo code is shown in Figure 1.

```
Relax(u,v , w_uv, Criterion)

Criterion: (Temp_const, Res_min), (Res_const, Temp_min)
If (Evaluate(Criterion) == true)
    Update (switching activity (w_uv),
    Update temperature (w_uv )
     d[v] = d[u] + 1;
     parent[v] = u
```

Figure 1. Relaxation routine.

After all the vertices have been relaxed, we have paths, which represent permissible binding of the operations to a resource. The algorithm selects the longest path and assigns the operations to the same resource. Since we have a directed acyclic graph, relaxing on a topologically sorted list guarantees that relaxing a node does not involve relaxing its children in the same iteration For comparability graph $G(V, E)$, $E = O(V^2)$ and the cost of relaxation of all the vertices is $O(E) = O(V^2)$. The vertices in the selected path are removed from the comparability graph and a new comparability graph is built with the remaining vertices and the above steps are repeated. The pseudo code of the Temperature-Aware binding algorithm is shown in Figure 2.

Since we relax vertices with minimum temperature rise, the operations on the longest path may not include all compatible operations as required by the resource constraint. Thus the initial binding might violate resource constraint. Then we perform post processing to meet the resource constraint. The post-processing algorithm tries to merge operations from the resource, which have minimum number of operations into other compatible resources. The final choice of the binding is determined such that it entails minimum rise of temperature. Post-processing iteratively decreases the resource number such that it meets the resource constraint. Obviously the constraint has to be at least equal to the feasible number of resources required to bind all the operations, which can be obtained by Left Edge algorithm. For temperature constraint resource minimizing involves deleting least populated resources and binding operations such that the temperature constraint of the resource (where the operation will be bound) is not violated. The overall flow is shown in Figure 3.

```
Input: Scheduled Data Flow Graph(DFG)
       Switching activity between the operations
       Criterion: (Temp_const, Res_min), (Res_const, Temp_min)

Begin:
  For each operation type
    While all the operations not bound to a resource
      Build comparability graph for the operation
      Label the edges with the switching activities
       Initialize the nodes
       Topologically sort the comparability graph
       For each vertex u taken in a topologically sorted order
          For each vertex v adjacent to u
            do Relax(u, v, wuv, Criterion)
          end For
       end For
       Pick the longest criterion safe path and bind
       the operations to a resource
       Increase the resource count of the operation type
       Delete the operations bound to a resource from the
       comparability graph
     end While
  end For
End

Output: Maximum temperature of Resource
        Number of resources required by each optype
```

Figure 2. Temperature-aware binding algorithm: Based on the criterion parameter, either temperature constrained resource minimization or resource constrained temperature minimization is performed.

```
Criterion Selection for Resource Binding
Temperature Aware Resource Binding (Criterion)
Use Left Edge Algorithm to find Feasible Resource size
if(Resource Constraint => Feasible Resource size)
     Post Processing the binding to satisfy Resource constraint
     with minimum Temp rise
Else
     Post Processing the binding to minimize resources under
     Temp constraint
```

Figure 3. General temperature-aware binding methodology.

4. Experimental Results

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

4.1 Experimental Setup

Figure 4 depicts our experimental flow. We have used benchmarks from two sources: applications from the MediaBench suite [14] and Data Flow Graphs (DFGs) of some popular DSP applications that are widely used in the high-level synthesis community. Using the SUIF and Machine-SUIF compiler infrastructure [21] we have extracted the DFGs of representative functions from MediaBench applications. The input DFGs were then scheduled using our own resource-constrained latency minimizing scheduler [17]. The input DFGs have been simulated to generate switching probabilities for individual operations using a trace of 10,000 input values. Functional modules used (our resource sets contained ALUs to execute add, subtract, and logical operations, and multipliers) have been synthesized using Synopsys Design Compiler onto the 180nm tsmc library. Capacitance values of modules have thereby been extracted to estimate switching power and this information was combined with bit toggle probabilities obtain through simulation to obtain switched capacitance values. Compatibility graphs for each resource type for the scheduled DFGs have then been created where edge weights are equal to the switched capacitances obtained as explained above. The compatibility graphs are given as input to the binding stage.

We generated binding solutions using three approaches. We call the first approach *Switching_Optimized* (SW_OPT in short). This is the binding scheme based on the min-cost network flow formulation proposed by Chang and Pedram [4]. They applied this formulation to low power register binding where the binding problem is formulated as a minimum cost clique covering problem, and solved it optimally using a transformation from max-cost flow algorithm to min cost flow. We have solved the network flow formulation using a software package developed by Goldberg [8].

The second binding solution was generated using our temperature-aware technique, where the same resource constraint that was used in the min cost flow approach has been used. We call this binding *Resource_Constrained_Temperature_Minimized* (RC_TEMP_MIN). The third solution was obtained by our temperature-aware technique after relaxing the resource constraint. In this case we determine the allocation, i.e., number of resources to be used, as a well as the actual binding to resources. We call this binding *Temperature_Constrained_Resource_Optimized* (TC_R_MIN).

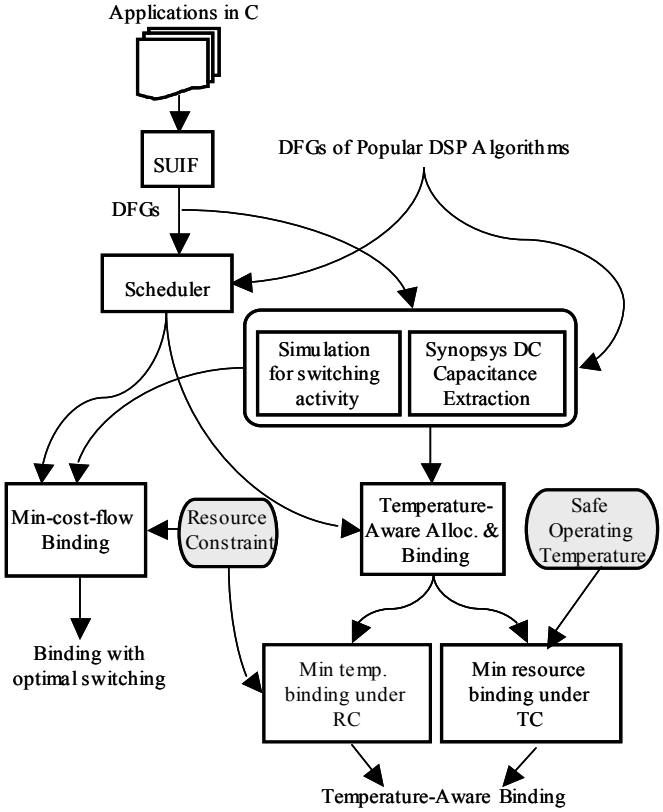

Figure 4. Overall experimental flow.

4.2 Results

Our first set of results present the maximum temperature any modules reaches in each benchmark. We have estimated the temperature levels for individual modules in each design using our

model described in Section 3.1. We took measurements with this model for an execution interval of 1000 clock cycles at a clock frequency of 100 MHz. Figure 5 depicts the maximum temperature reached by the ALUs. Figure 6 presents the maximum temperature reached by the multipliers for the three binding solutions. (jctrans_2 did not contain any multiplication.) TC_R_MIN binding always yields the best solution. It reduces the maximum observed temperature by as high as 19.6°C, and 7.6 °C on average for multipliers and 11.9°C for ALUs. However, often times this comes at the expense of increased number of resources. Table 1 shows the number of resources used by SW_OPT binding and TC_R_MIN binding. We performed the RC_TEMP_MIN binding by using the same resource constraint that was used for the SW_OPT binding. In that case, the maximum temperature on the resources increases with respect to the TC_R_MIN binding. However, it is still below SW_OPT solution. Particularly, it reduces the maximum temperature by as much as 10.3°C and 11.2°C, for multipliers and ALUs, respectively (2.7°C and 3.6°C on average).

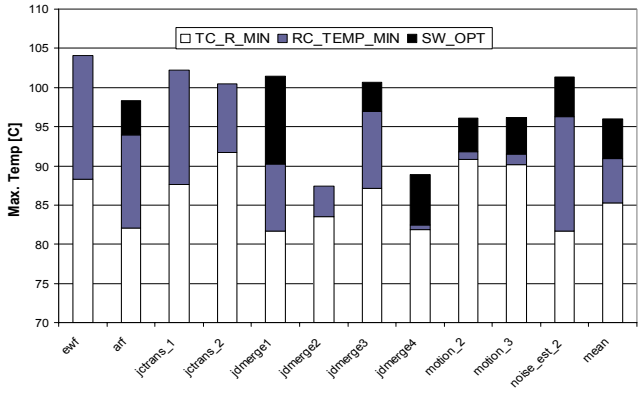

Figure 5. Maximum temperature reached by ALUs.

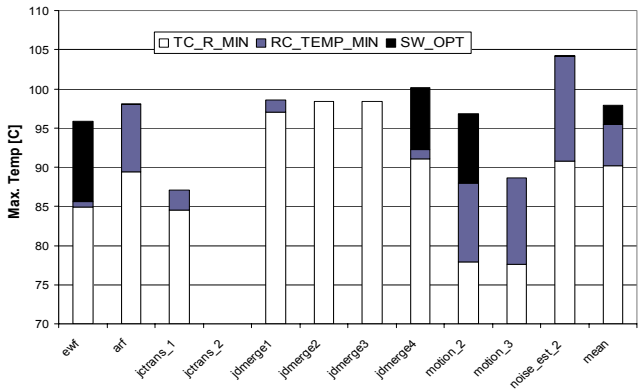

Figure 6. Maximum temperature reached by multipliers.

When we compared the impact of our binding techniques onto the power consumption, we realized that control of maximum temperature comes at the expense of some power overhead. For instance, the RC_TEMP_MIN binding uses a larger number of resources to minimize maximum temperature. This in turn causes the total power consumption of the design to increase. We will present results to this end shortly. For the moment, assuming that there is some overhead associated with temperature controlled binding, we can view TC_R_MIN and RC_TEMP_MIN as binding schemes with different levels of aggressiveness in terms of temperature optimization. If hot spot prevention is of highest priority due to the stringent reliability requirements one of the two temperature-aware binding schemes would be chosen. Whether to use TC_R_MIN or RC_TEMP_MIN would depend on the tolerable power overhead. If only a very small overhead is tolerable then RC_TEMP_MIN would be chosen, which still keeps the temperature of the hottest module below the levels of SW_OPT. If the power overhead allowance can be relaxed further then the maximum reduction in the temperature of the hottest resource can be achieved using TC_R_MIN.

Table 1. Resource requirements for SW_OPT and TC_R_MIN bindings.

	SW_OPT [MUL, ALU]	TC_R_MIN [MUL, ALU]
Ewf	3, 5	4, 8
Arf	4, 2	5, 4
jctrans_1	2, 3	2, 7
jctrans_2	0, 4	0, 6
jdmerge1	3, 6	3, 7
jdmerge2	3, 6	3, 9
jdmerge3	3, 6	3, 9
jdmerge4	3, 5	5, 9
motion_2	4, 6	6, 8
motion_3	4, 6	6, 8
Noise_est_2	3, 4	4, 7

Now, we present the actual power overhead associated with TC_R_MIN and RC_TEMP_MIN with respect to SW_OPT binding. Figure 7 illustrates the relative increase in the leakage power consumed by all resources in a benchmark for TC_R_MIN and RC_TEMP_MIN bindings with respect to the normalized value of SW_OPT.

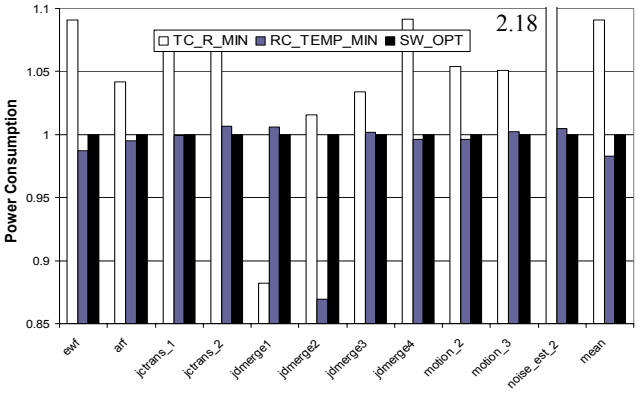

Figure 7. Normalized leakage power consumption of the three techniques.

For example, a value of 0.86 corresponds to a 14% reduction in leakage power consumed by all resources in a benchmark with respect to SW_OPT binding. We see that the leakage power for TC_R_MIN binding ranges from 0.88 to 2.18 with an average overhead of 1.09 (9% increase over SW_OPT). Although TC_R_MIN achieves minimal temperatures, it increases the number of resources used significantly. Hence, the total leakage power tends to increase. In the case of RC_TEMP_MIN, the leakage power ranges from 0.86 to 1.01 of SW_OPT, reducing the leakage to 0.98 (2% reduction) on average. In almost all cases RC_TEMP_MIN achieves a better leakage behavior. Although RC_TEMP_MIN successfully reduces the maximum temperature

(hot spot) it generates a few other resources with relatively high temperatures. This in turn can increase the leakage power.

Figure 8 depicts the total (dynamic+leakage) power consumption for the benchmarks. TC_R_MIN incurs an overhead in the range 14% to 138% with respect to SW_OPT. The average overhead is at 1.34 (34% increase in total power). The total power of the RC_TEMP_MIN binding ranges between 0.90 and 1.43 of the SW_OPT, yielding a maximum overhead of 43%. The average overhead of RC_TEMP_MIN on total power is 5%.

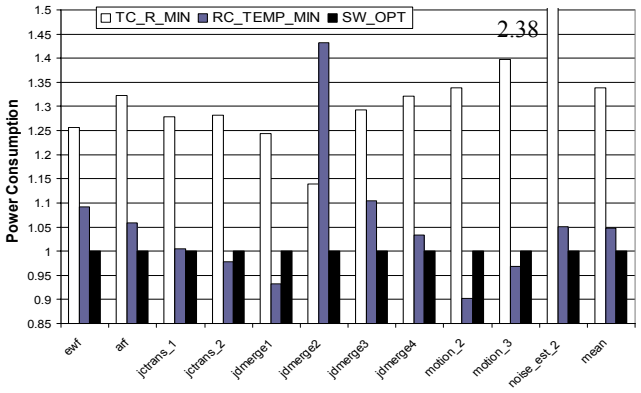

Figure 8. Normalized total power consumption of the three techniques.

For future technologies, we would expect the dominance of leakage power to increase significantly. Although it is not a straightforward task to project all of our temperature and power models onto a future technology scale, a rough estimation based on the exponential trend in leakage vs. temperature relationship indicates that the power overhead of our binding scheme will incur a continuously diminishing power overhead. The power overhead of our techniques mostly originates from the fact that we do not give priority to optimizing the switching activity on the resources. As the contribution of switching power to the total power consumption decreases with technology scaling we believe that our techniques will be able to minimize the total power consumption.

Our results show that our temperature-aware binding technique effectively controls the hot spot occurrences. If there are hard constraints on the temperature, but not on the area, then the TC_R_MIN approach can be used to keep the chip temperature below a given constraint with a small impact on area and total power consumption. If on the other hand, the area is highly constrained RC_TEMP_MIN approach would leverage reliability.

5. CONCLUSIONS

We have introduced resource binding techniques to create awareness of temperature effects during high-level synthesis. Our main goal was to effectively minimize the maximum temperature that is reached by any module in a design. A reliability-driven design methodology can leverage on this mechanism to prevent or reduce the likelihood of hot spots on a chip.

Our results show that we can bound the maximum temperature on any module with overhead on area (28% increase in number of multipliers and 54% increase in the number of ALUs), and power (34% increase in total power) using the temperature constrained resource minimization (TC_R_MIN) technique. Using the resource constrained temperature minimization (RC_TEMP_MIN) technique, on the other hand, the maximum observed temperature can be reduced by 5^{o}C on average, while incurring no area and a small (5% on average) power penalty.

6. REFERENCES

1. Basu, A., et al. Simultaneous Optimization of Supply and Threshold Voltages for Low-Power and High-Performance Circuits in the Leakage Dominant Era. *Design Automation Conference*. 2004.

2. Brooks, D. and M. Martonosi. Dynamic Thermal Management for High-Performance Microprocessors. *International Symposium on High-Performance Computer Architecture*. 2001.

3. Cao, L., et al., *Transient Thermal Management of Portable Electronics using Heat Storage and Dynamic Power Dissipation Control.* IEEE Transactions on Components, Packaging, and Manufacturing Technology-Part A, 1998. **21**(1): p. 113-123.

4. Chang, J.M. and M. Pedram. Register Allocation and Binding for Low Power. *Design Automation Conference*. 1995.

5. Chang, J.M. and M. Pedram. Module Assignment for Low Power. *European Design Automation Conference*. 1996.

6. Chu, C.C.N. and D.F. Wong. A Matrix Synthesis Approach to Thermal Placement. *International Symposium on Physical Design*. 1997.

7. Cong, J., J. Wei, and Y. Zhang. A Thermal-Driven Floorplanning Algorithm for 3D ICs. *International Conference on Computer-Aided Design*. 2004.

8. Goldberg, A.V., *An Efficient Implementation of a Scaling Minimum-Cost Flow Algorithm.* Journal on Algorithms, January 1997. **22**(1): p. 1--29.

9. Gunther, S., et al., *Managing the Impact of Increasing Microprocessor Power Consumption.* Intel Technology Journal, 2001.

10. Huang, W., et al. A Framework for Dynamic Energy Efficiency and Temperature Management. *International Symposium on Microarchitecture*. 2000.

11. Huang, W., et al. Compact Thermal Modeling for Temperature-Aware Design. *Design Automation Conference*. 2004.

12. Intel Corp. Intel® Itanium® Processor Overview, www.intel.com/design/itanium/itanium/

13. Krum, A., *Thermal Management*, in *The CRC Handbook of Thermal Engineering*, F. Kreith, Editor. 2000, CRC Press: Boca Raton, FL.

14. Lee, C., M. Potkonjak, and W.H. Mangione-Smith. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communicatons Systems. *International Symposium on Microarchitecture*. 1997.

15. Liao, W., F. Lei, and L. He. Microarchitecture Level Power and Thermal Simulation Considering Temperature Dependent Leakage Model. *International Symposium on Low Power Electronics and Design*. 2003.

16. Lyuh, C. and T. Kim, *High-level Synthesis for Low Power Based on Network Flow Method.* IEEE Transactions on Very Large Scale Integration Systems, June 2003. **1**(3).

17. Memik, S.O., et al. A Super-Scheduler for Embedded Reconfigurable Systems. *International Conference on Computer-Aided Design*. 2001.

18. Sabry, M.N. Dynamic Compact Thermal Models: An Overview of Current and Potential Advances. *International Workshop on Thermal Investigations of ICs and Systems*. 2002.

19. Skadron, K., T. Abdelzaher, and M.R. Stan. Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management. *Eighth International Symposium on High-Performance Computer Architecture*. 2002.

20. Skadron, K., et al. Temperature-aware Microarchitecture. *International Symposium on Computer Architecture*. 2004.

21. Stanford University Compiler Group The SUIF 2 Compiler System, http://suif.stanford.edu/suif/suif2/

22. Tsai, C. and S. Kang. Standard Cell Placement for Even On-Chip Thermal Distribution. *International Symposium on Physical Design*. 1999.