Peak Temperature Control and Leakage Reduction During 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

Temperature is becoming a first rate design criterion in ASICs due to its negative impact on leakage power, reliability, performance, and packaging cost. Incorporating awareness of such lower level physical phenomenon in high level synthesis algorithms will help to achieve better designs. In this work, we developed a temperature aware binding algorithm. Switching power of a module correlates with its operating temperature. The goal of our binding algorithm is to distribute the activity evenly across functional units. This approach avoids steep temperature differences between modules on a chip, hence, the occurrence of hot spots. Starting with a switching optimal binding solution, our algorithm iteratively minimizes the maximum temperature reached by the hottest functional unit. Our algorithm does not change the number of resources used in the original binding. We have used HotSpot, a temperature modeling tool, to simulate temperature of a number ASIC designs. Our binding algorithm reduces temperature reached by the hottest resource by 12.21°C on average. Reducing the peak temperature has a positive impact on leakage as well. Our binding technique improves leakage power by 11.89%, and overall power by 3.32% on average at 130nm technology node compared to a switching optimal binding.

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: Leakage, Binding, Temperature, Switching.

1. INTRODUCTION

The continuous technology scaling is allowing the designers to place more transistors per unit silicon area and pack more and more transistors in a single chip. For example, Intel Itanium 2 processor contains more than 200 million transistors [17]. The increase in number of transistors per unit silicon area is achieved at the expense of increased power densities. Power density in microprocessors has surpassed that of a kitchen's hot plate at 0.6um technology [4]. The trend continues to increase as technology scales following Moore's law, which is not expected to slow down for at least another decade. One of the most important consequences of this power increase is its effect on temperature. Temperature of a chip increases in proportion to the power consumption. In addition, since different parts of a chip have different levels of activity, we are faced with large operating temperature variations. Regions on a chip that generate excessive amounts of heat and consequently reach high temperatures are referred to as "hot spots".

Temperature has a significant impact on circuit performance. Increase in temperature decreases carrier mobility, hence, reduces switching speed of transistors. Interconnect resistance increases with temperature as well. Hot functional units can exhibit timing violation and lead to functional incorrectness of the circuit. Circuit reliability is also heavily impacted by temperature. 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. Further, leakage has exponential dependence on temperature. As leakage power becomes dominant in the present and future technologies, high temperatures can cause substantial increase in total power consumption of the chip. Also rising cooling costs will limit electronics industry's ability to develop commercial systems.

Present day design flows optimize different design metrics such as power, area, and delay. Previous work has shown that design planning and optimization above the physical synthesis stage can aid reaching design closure efficiently [6]. Similarly managing temperature at higher stages of the design flow should be beneficial to achieve the desired temperature optimizations. In this paper, we will concentrate on a binding scheme that aims to minimize the occurrence of hot spots. Note that temperature optimization is not always equivalent to power optimization. Localized heating occurs at a much faster pace than chip wide heating due to the slow rate of lateral heat propagation [21]. This creates different temperatures at different locations in a chip. Therefore, minimizing the total power consumption does not directly translate to decreasing the hot spots.

In this paper, we present techniques to incorporate temperature awareness into high level synthesis. If tasks such as scheduling, resource allocation, and binding, which impact the activity of functional resources, are performed with such awareness, temperature increase can be controlled more effectively. Our specific contributions in this paper are as follows:

- We formulate the temperature aware binding problem.
- We develop a temperature aware binding algorithm, which minimizes the maximum temperature reached by the hottest functional unit without increasing the number of resources.

ISLPED'05, August 8–10, 2005, San Diego, California, USA.

- We evaluate the impact of temperature on leakage power. We compare the power consumption of switching optimal binding and temperature aware binding algorithms and show the possible saving in leakage and overall power for 130nm technology node.
- We present projected savings in leakage and overall power consumption for 100nm generation using our temperature aware binding.

In this work, we chose to handle the problem at a high level while maintaining a modular design flow. Incorporating temperature awareness into other steps such as scheduling and developing combined feedback-driven techniques could bring further benefits. While such a combined effort would further improve effectiveness, we believe that our experimental results show that our approach is in itself effective to control the temperature increase.

The remainder of this paper is organized as follows. Section 2 gives an overview of related work. In Section 3.1 we discuss the switching optimized binding for low power using flow formulation. In Section 3.2 we present our temperature aware binding algorithm. Section 4 presents our experimental flow and results. We conclude with a summary in Section 5.

2. RELATED WORK

Gunther et al. [14] have shown a non-linear relationship between the cooling capabilities and the cost of the solution as power increases. This shows the importance of limiting the maximum temperature and thus power consumption for the electronics industry to successfully deploy commercial systems. For design and analysis of high performance microprocessors various techniques to model thermal effects have been developed [16, 19, 20]. Runtime thermal management via clock gating using real-time temperature sensing has been included in Intel Pentium 4 processors [14]. Other techniques such as frequency and/or voltage scaling, sub-banking, etc. have been investigated [5, 15]. Skadron et al. [21] proposed a thermal model HotSpot for the architectural level. The authors also proposed several dynamic thermal management methods such as frequency scaling, localized toggling, and migrating computation to spare hardware units.

At the other spectrum are the physical design tools to enable even thermal distribution on chips. Tsai and Kang developed a standard cell placement tool for even on-chip thermal distribution [23]. Chu and Wong proposed a matrix synthesis approach to thermal placement [9]. Cong et al. introduced a thermal-driven floorplanning algorithm for 3-D ICs [10]. Basu et al. introduced the electrothermal energy-delay-product optimization scheme to perform simultaneous optimization of supply and threshold voltages in CMOS circuits [3]. Thermal models for interconnects have presented in [1, 8]. Banerjee et al. [2] presented a methodology for making temperature and reliability aware power/performance/cooling-cost tradeoffs in leakage dominant ICs at the circuit level.

As power dissipation is converted to heat, controlling the die temperature will be essential for performance and leakage control for microprocessors [4]. Reducing temperature will be equally important for cost effective cooling solutions in ASICs. In this work we aim to accomplish this goal by developing a temperature aware binding algorithm to be integrated within high level synthesis.

3. BINDING PROBLEM

The binding problem takes as input a scheduled Data Flow Graph (DFG) and fixed number of functional units. It then assigns operations to those resources based on different objectives. In the next section we will describe a low power binding algorithm that minimizes the total switching capacitances of the resources.

3.1 Low Power Binding

Low power binding assigns operations to functional units such that the total switching capacitance of the resources is minimized. This can also include choosing among different architectures of the resources for minimizing power. Consider a set of *m* resource types $\{R_1 \ldots R_m\}$. First, a comparability graph G_C is built from a scheduled DFG for each resource type. The vertices of G_C represent the operations that a particular resource type can execute. A comparability graph is essentially a compatibility graph with transitive orientation. There exists a directed edge e_{ij} between two operations v_i and $v_j \in G_C$ if start time of $(v_j) \geq$ finish time of (v_i). An edge e_{ij} has a weight s_{ij} which is the estimated switched capacitance if operations v_i and v_j are bound to the same resource consecutively. Chang and Pedram [7] formulated the resource constrained low power binding scheme as min-cost network flow problem. We will refer to this binding scheme as the *Flow based Binding (FB)* in the remainder of our discussion. 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. A low power binding heuristic for fixed number of functional units has also been proposed by Davoodi and Srivastava [11].

3.2 Temperature Aware Binding

The flow based binding can bind uneven number of operations to different functional units in pursuit of its objective of minimizing the total switched capacitance. This in turn gives minimal overall switching power. As a result, some functional units can end up switching more than others, dissipate more power, and become hot spots. The uneven power densities between different resources of the same type (and different types) lead to large variation in operating temperature at different locations in the chip. Our *Temperature Aware Binding (TB)* algorithm distributes operations to functional units aiming to create an even power dissipation across resources. This helps us control the rate at which the temperature of individual resources increases and prevents some resources from reaching disproportionately high temperatures with respect to others. The main idea behind our approach is to reassigning operations from high switching resources to low switching resources. Starting with a switching optimal binding solution our temperature aware binding iteratively improves the temperature profile while keeping the resource constraint unchanged.

We will describe our binding algorithm in Section 3.2.2 in more detail. Let us first present a motivational example to illustrate the possible impact of our temperature aware technique.

3.2.1 Motivational Example

Let us consider the binding of multiplication operations in a DFG onto five multipliers (MUL_1, …, MUL_5). Figure 1 presents the power consumption of the five multipliers after flow based and temperature aware binding. After flow based binding the maximum power dissipated by the hottest module MULT_5 is 260 mW. The temperature aware binding algorithm reduces the maximum power dissipation of this module to 230mW although switching power increases for MULT_1 and MULT_2. We observe that the difference between the maximum power consumption (MUL_5) and the minimum (MUL_1) is less for temperature aware binding, i.e., our technique creates a more even power distribution. In Figure 2, we show the temperature profile of the multipliers. It is evident that the temperature profile of the resources follows a pattern similar to their power dissipation trends. We observe that for flow based binding MUL_1 (the coolest resource) reaches 73.47°C, but MUL_5 (the hottest resource) reaches the maximum temperature of 122.14°C. Although our temperature aware binding algorithm increased the temperature to 83.06°C from 73.47°C (MUL_1), it reduced the maximum temperature from 122.14°C to 113.36°C (MUL_5).

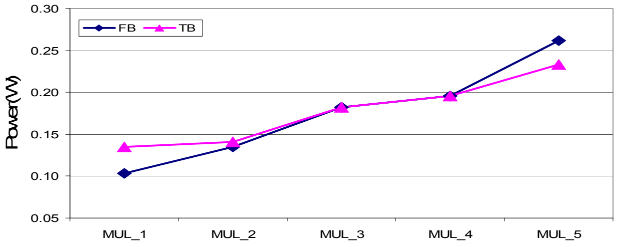

Figure 1. Power profile across the multipliers. The temperature aware binding achieves a more even distribution of power consumption across resources.

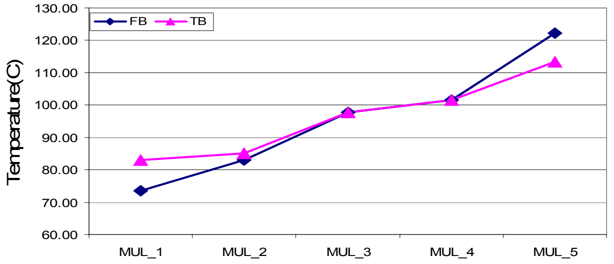

Figure 2. Temperature profile across the multipliers. The highest temperature reached by any resource has been decreased from 122.14 °C (FB) to 113.36 °C (TB).

The switching power and the temperature of the hottest resource (in this example module MUL_5 is the hottest resource) are shown in Table 1 for flow based and temperature aware binding.

Table 1. Power and temperature of the hottest resource for flow based and temperature aware binding.

	Switching Power (W)		Temperature (°C)	
Binding	FB	TB	FB	TB
MUL_5	0.26	0.23	122.14	113.36

We will present our detailed analysis in Section 4.2, which shows that our binding algorithm succeeds in limiting the highest temperature reached by the hottest resource effectively. We will also present savings in leakage power due to reduction of the maximum temperature and reduction in total power. Although we see that our algorithm increases switching and hence the total dynamic power, due to reduction in leakage power we generally observe an overall improvement in total power. This shows that traditional power minimization techniques might result in suboptimal designs when temperature is not taken into account.

3.2.2 Temperature Aware Binding Algorithm

In this section we discuss our temperature aware binding algorithm. The input to our algorithm is an initial binding solution obtained using min cost flow formulation [7]. Our algorithm iteratively transforms this solution by moving operations across resources such that activity among functional units is distributed evenly. We refer to the rebinding of an operation from one functional unit to another as a *move*. We represent the start time and finish time of operation i as $St(i)$ and $Ft(i)$ respectively. We define two types of moves – *insert* and *swap*.

Let us consider operations i, j, and k initially assigned to functional unit R_h. The total switched capacitance of resource R_h is $\{s_{ij} + s_{jk}\}$. Operation j can be moved to another functional unit R_l and placed in between two operations x and z if $Ft(x) \leq St(j) < Ft(j) \leq St(z)$. Operation j is then *compatible* with operations x and y and with the resource R_l. We call this type of move *insert* and an example of this move is shown in Figure 3(a). Insert decreases the switched capacitance of R_h from $\{s_{ij} + s_{jk}\}$ to $\{s_{ik}\}$ and increases that of R_l from $\{s_{xz}\}$ to $\{s_{xi} + s_{jz}\}$.

However, there can exist an operation y between operations x and z on R_l such that $St(y) \leq St(j) \leq Ft(y)$ or $St(y) \leq Ft(j) \leq Ft(y)$. In this case operation j conflicts with operation y. Operation j can be moved from R_h to R_l only if y is compatible with operations i and k and can be simultaneously moved to R_h. We call this move as *swap*. Figure 3(b) illustrates an example of a *swap* move. The total switched capacitance of R_h changes from $\{s_{ij} + s_{jk}\}$ to $\{s_{iy} + s_{yk}\}$ and that of R_l from $\{s_{xy} + s_{yz}\}$ to $\{s_{xj} + s_{jz}\}$.

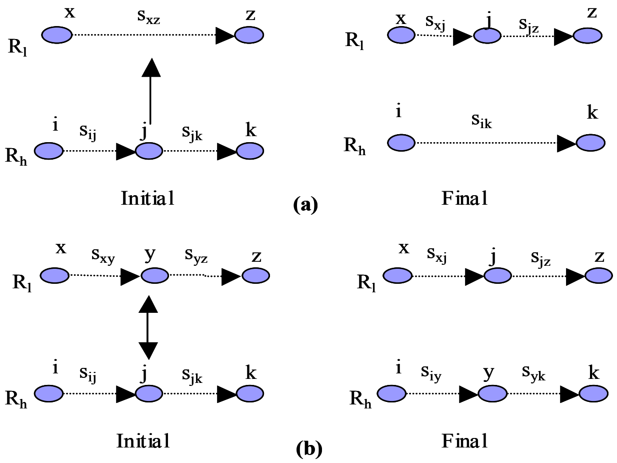

Figure 3. Moves in Temperature Aware Binding Algorithm. (a) Insert move: moves one operation onto a new resource. (b) Swap move: swaps two operations between two resources.

The *insert* move can reduce the switched capacitance of the source functional unit R_h and increase the switched capacitance of the destination functional unit R_l. On the other hand, *swap* increases the switched capacitances of both functional units. Note that the min cost flow formulation is optimal in minimizing the total switched capacitance in a single iteration of a scheduled DFG. We refer to this as the intra-iteration switched capacitance. However, the optimality condition does not hold when the input DFG is a representation of a computation within a loop body. In other words, this approach does not optimally minimize data transitions in cyclic execution, i.e., inter-iteration switched capacitance is not minimized.

The *swap* and *insert* moves can have different consequences depending on whether we take the intra-iteration switched capacitance into consideration or not. If we do not consider inter-iteration switching, then both *insert* and *swap* are very likely to increase the total intra-iteration switched capacitance. Note that since we started with an optimal intra-iteration switched capacitance binding, our moves are expected to degrade this optimal solution. However, we may improve the inter-iteration switched capacitance (by either *insert* or *swap*), which can even decrease the overall switched capacitance. We will discuss the impact of our technique onto the total switching activity, hence, the switching power in Section 4.2 in detail.

Our temperature aware algorithm operates as follows. First, the resources with the highest and the lowest total switched capacitances – R_{max} and R_{min} are selected. Our algorithm identifies operations that can be moved from R_{max} to R_{min} using one of the two moves. The *swap* move would imply that while one operation is moved from R_{max} to R_{min}, another operation is moved from R_{min} to R_{max}. This would be an inherently bad move for the intra-iteration switching of R_{max} since this would force the binding solution to deviate from the optimal. There is one possible exception to this. For the *first* and the *last* operation bound to R_{max}, we allow *swap* only if it improves the inter–iteration switching compared to flow based binding. For any other operation bound to R_{max} our algorithm prohibits *swap*.

Let us consider moving an operation *j* from R_{max} to R_{min}. (This could be the result of either a *swap* or an *insert*.) Assume that moving *j* to R_{min} leads to a decrease of switched capacitance of R_{min} by amount D_j^{min} (if this move improves the inter-iteration switching) and that of R_{max} by D_j^{max}. Then, among all operations $\in$ R_{max} compatible with R_{min} the operation which leads to *maximum* $|D_j^{max} + D_j^{min}|$ will be selected for the move.

If such a move leads to an increase in the switching activity of R_{min} by I_j^{min}, then, among all operations $\in$ R_{max} compatible with R_{min} we move the operation *j* for which we have *minimum* $|I_j^{min} - D_j^{max}|$.

After one move, the total switched capacitance of R_{max} will decrease. The switched capacitance of R_{min} could decrease if the move improves the inter-iteration switching. Otherwise, the switched capacitance of R_{min} would increase. After the move, our algorithm evaluates the switched capacitances of all resources and identifies the new pair of R_{max} and R_{min} . The algorithm terminates once the difference between the total switched capacitances of the current R_{max} and R_{min} is within a pre-defined range. We have experimentally determined that a threshold value of 20mW yields the best results.

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

We have used benchmarks from two sources: applications from the MediaBench suite [18] and DFGs of some common DSP applications. Using the SUIF and Machine-SUIF compiler infrastructure [22] we have extracted the DFGs of representative functions from MediaBench applications. The input DFGs were then scheduled using a list scheduler. The input DFGs have been simulated to generate switching probabilities for individual operations using a trace of 10,000 input values. Functional modules (our resource sets contained ALUs to execute add, subtract, and logical operations and multipliers) have been synthesized using Synopsys Design Compiler onto a 180nm technology library. We used scaling trends [4] to get switched capacitance and nominal power values for a switching activity of 0.5 at 130nm technology node for each resource type. Capacitance values of modules have thereby been extracted to estimate switching power and this information was combined with bit toggle probabilities obtained through simulation to generate switched capacitance values. Comparability 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 comparability graphs are given as input to the binding stage.

We first generated binding solutions using flow formulation as proposed by Chang and Pedram [7] based on the min-cost network flow formulation. We have solved the network flow formulation using a software package developed by Goldberg [13]. The generated binding using min-cost network flow is then used as an input to our temperature aware binding algorithm. We calculate the total switched capacitance S_R^{FB} of each resource R due to operations bound to it by flow based binding (FB) and the total power P_R^{FB} dissipated by the resource using the nominal power values for a switching activity of 0.5. Similarly we calculate the total switched capacitance S_R^{TB} of each resource R due to operations bound to it by temperature aware binding (TB) and the total power P_R^{TB} dissipated by the resource.

We have used HotSpot [21] 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 instantaneous power values of each functional unit are input in the form of a trace file where each line corresponds to a sampling interval. Finally, HotSpot simulates the activity on the chip and computes the steady state temperatures of the functional units.

We have assumed the same chip-packaging configuration as modeled by HotSpot. 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 Figure 4.

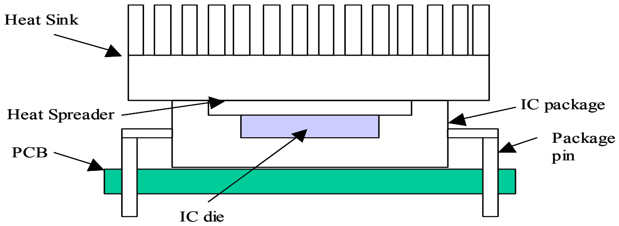

Figure 4. Side view of a typical chip package.

A single thermal resistance $R_{convection}$ 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 shown in Figure 5. 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.

$C_{convection}$ = 140.4 J/K	$R_{convection}$ = 0.2 K/W
Heat sink side = 4 mm	Heat sink thickness = 6.9 mm
Spreader side = 2 mm	Spreader thickness = 1 mm
Chip thickness = 0.5mm	Sampling interval = 3.33 us

Figure 5. HotSpot parameters.

We generate the power trace file for HotSpot using the values P_R^{FB} and P_R^{TB} for each functional unit for the flow based and the temperature aware binding algorithm, respectively. We set the initial temperatures of each functional unit at 77°C and the heat spreader and heat sink at 47°C. Further, we implemented a simulated annealing based floorplanner to generate floorplans for our benchmarks. The power trace file and the floorplans are used as input to HotSpot, which computes the temperature Γ_R^{FB} and Γ_R^{TB} of resource R for the flow based binding algorithm and our algorithm respectively. Our overall experimental flow is shown in Figure 6.

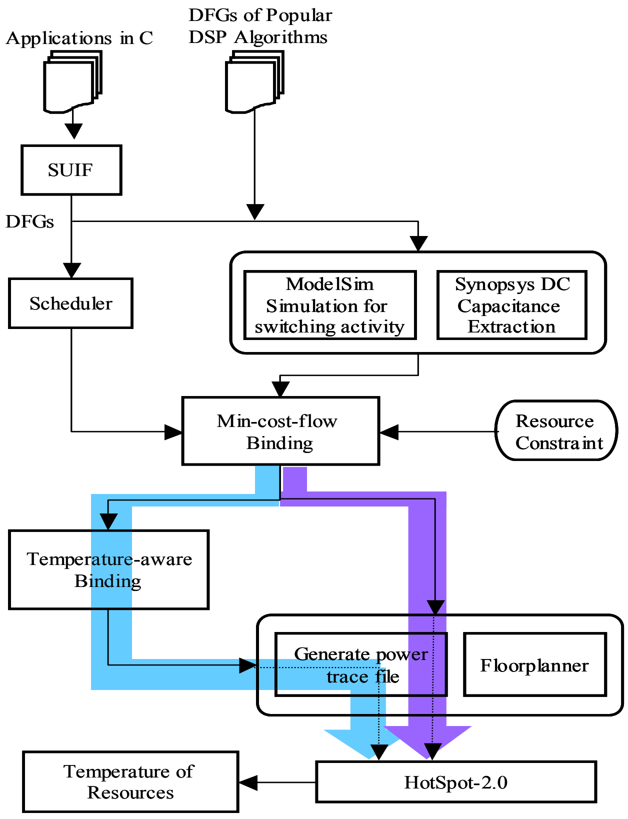
Applications in C
DFGs of Popular DSP Algorithms
SUIF
DFGs
Scheduler
ModelSim Simulation for switching activity
Synopsys DC Capacitance Extraction
Min-cost-flow Binding
Resource Constraint
Temperature-aware Binding
Generate power trace file
Floorplanner
Temperature of Resources
HotSpot-2.0

Figure 6. Experimental flow.

4.2 Results

In this section we discuss the experimental results. Table 2 shows the number of resources – multipliers and ALUs used by our benchmarks. Note that *jctrans_2* does not have any multiplication operations.

Table 2. Resource requirements across benchmarks.

	arf	jctrans_1	jctrans_2	jdmerge_1	jdmerge_2	jdmerge_4	motion_2	motion_3	noise_est2	fdct	fft
MUL	4	2	-	3	3	3	3	3	3	5	4
ALU	3	3	4	5	6	4	5	6	4	5	4

4.2.1 *Temperature Reduction*

In Figure 7 we show the difference between the maximum temperatures reached by the hottest resource in the flow based binding and our binding algorithm. In other words, we have plotted Γ_{diff} = $(\Gamma_R^{FB})_{max}$ – $(\Gamma_R^{TB})_{max}$ for the hottest resource in each benchmark. We obtain a maximum Γ_{diff} of 19.86°C for *jdmerge_4* benchmark. Overall we have obtained an average Γ_{diff} of 12.21°C across all benchmarks. Thus by decreasing the steady state temperatures of the functional units, we can effectively reduce the possibility of occurrence for "hot spots" in a design.

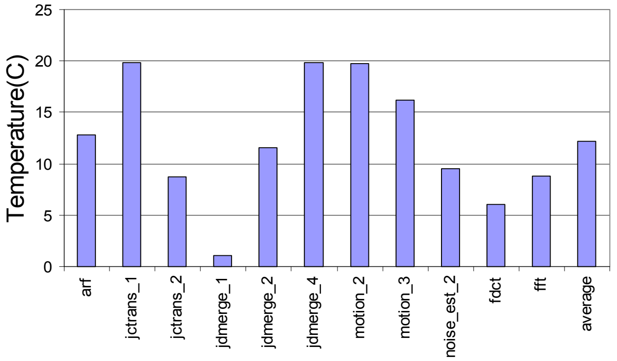

Figure 7. Reduction in temperature of the hottest resource in each benchmark.

4.2.2 *Effect of Temperature on Leakage*

We have evaluated the impact of temperature on leakage power using the relation between temperature and leakage current as well as the relation between temperature and leakage power [12] at 130nm and 100nm technology nodes. From this data, we derived the relationship between temperature and leakage power.

We report reduction in leakage with respect to flow based binding. Figure 8 presents the leakage power savings using our temperature aware binding. We present three values for each benchmark. The first two bars indicate the leakage savings for individual resource types (multipliers and ALUs) and the last bar is the total leakage saving. For multipliers and ALUs we get an average saving of 8.83% and 11.91%, respectively. The average of total leakage saving is 11.89%.

The negative values indicate cases where leakage power is increased with respect to flow based binding. For example, for the *noise_est_2* benchmark, although we decrease the maximum temperature by 9.52°C, we increase the average temperature of the multipliers and ALUs resulting in an increase of 2.18% in total leakage over flow-based binding. For the *fdct* benchmark we decrease the multiplier peak temperature and leakage power by 8.77°C and 9.63% respectively. Although peak temperature of its ALUs deceases, the average temperature increases which results in 11.30% increase in ALU leakage power. The total leakage saving is 4.77%. Overall our algorithm reduced the leakage power of multipliers and ALUs by 8.83% and 11.91% resulting in 11.89% saving in the total leakage power. Note that the total percentage savings over all the resources is not a simple summation of individual percentage leakage savings.

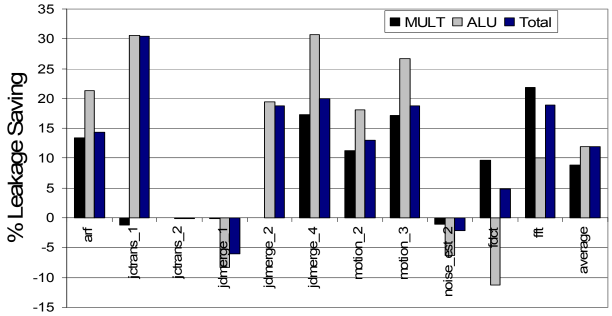

Figure 8. Percentage leakage power saving for different functional units.

Next, in Table 3 we present the total power savings using our temperature aware binding over flow based binding. We see that switching power increases on average by 2.16% due to our temperature aware binding. This is expected since our temperature aware algorithm starts with switching optimal binding and iteratively tries to reduce the maximum power dissipation of the resource set, which can lead to perturbation of the optimal switching. However, for some benchmarks we improve switching power by reducing inter-iteration switching as discussed in Section 3.2. For example *fft* shows 3.58% improvement in switching power. At 130nm node, the leakage power improvement is 11.89% and the total power improvement is 3.32%. We also projected the total power saving at 100nm node assuming that the same temperature profiles will be achievable. Switching power still increases by 2.16% but leakage power saving is 16.94% leading to a total power saving of 10.39% on average.

Table 3. Percentage improvement in leakage and total power at 130nm and 100nm technology nodes.

	%Change	% Imp at 130 nm		% Imp at 100 nm	
Benchmarks	Switching	Leakage	Total	Leakage	Total
arf	-5.21	14.39	1.67	16.15	7.02
jctrans_1	-2.16	30.44	5.88	28.57	12.62
jctrans_2	-5.28	-0.09	-4.07	-0.34	-3.11
jdmerge_1	-2.84	-6.01	-3.52	-4.37	-3.50
jdmerge_2	0.57	18.80	9.91	41.48	33.83
jdmerge_4	-4.01	19.94	6.36	28.56	17.86
motion_2	-0.30	13.07	6.52	23.56	17.52
motion_3	1.77	18.81	10.17	25.95	19.28
noise_est_2	-7.92	-2.18	-6.23	-0.22	-3.98
fdct	-1.93	4.77	0.06	5.11	1.67
fft	3.58	18.88	9.72	21.94	15.11
Average	**-2.16**	**11.89**	**3.32**	**16.94**	**10.39**

5. Conclusions

In this work, we presented a temperature aware binding algorithm for high level synthesis. Our algorithm tries to minimize the variation in power consumption across different functional units on a chip. This is achieved by performing a sequence of insert and swap moves on an initial binding, which is switching optimized. On average our technique reduces temperature reached by the hottest resource by 12.21°C. This in turn, leads to reduction in leakage power by 11.89% and a reduction in overall power consumption by 3.32% at 130nm technology node. On 100 nm generation designs, our binding algorithm reduces leakage by 16.94% and total power by 10.39%.

6. References

1. Banerjee, K., et al. On thermal effects in deep sub-micron VLSI interconnects. *Design Automation Conference*. 1999.
2. Banerjee, K., S.-C. Lin, and V. Wason. Leakage and variation aware thermal management of nanometer scale ICs. *IMAPS-Advanced Technology Workshop on Thermal Management*. 2004.
3. 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.
4. Borkar, S., *Design challenges of technology scaling.* IEEE Micro, July-August 1999. **19**(4): p. 23--29.
5. Brooks, D. and M. Martonosi. Dynamic thermal management for high-performance microprocessors. *International Symposium on High-Performance Computer Architecture*. 2001.
6. Cadence Whitepaper, Global Synthesis for Timing Closure, http://www.cadence.com/whitepapers/
7. Chang, J.M. and M. Pedram. Register allocation and binding for low power. *Design Automation Conference*. 1995.
8. Chiang, T.Y., K. Banerjee, and K.C. Saraswat, *Analytical thermal model for multilevel VLSI interconnects incorporating via effect.* IEEE Electron Device Letters, 2002. **23**(1): p. 31-33.
9. Chu, C.C.N. and D.F. Wong. A matrix synthesis approach to thermal placement. *International Symposium on Physical Design*. 1997.
10. Cong, J., J. Wei, and Y. Zhang. A thermal-driven floorplanning algorithm for 3D ICs. *International Conference on Computer-Aided Design*. 2004.
11. Davoodi, A. and A. Srivastava. Effective graph theoretic techniques for the generalized low power binding problem. *International Symposium on Low Power Electronics and Design*. 2003.
12. Fallah, F. and M. Pedram, *Standby and active leakage current control and minimization in CMOS VLSI circuits.* IEICE Transactions on Electronics, 2005. **E88-C**(4): p. 509-519.
13. Goldberg, A.V., *An efficient implementation of a scaling minimum-cost flow algorithm.* Journal on Algorithms, January 1997. **22**(1): p. 1--29.
14. Gunther, S., et al., *Managing the impact of increasing microprocessor power consumption.* Intel Technology Journal, February 2001.
15. Huang, W., et al. A framework for dynamic energy efficiency and temperature management. *International Symposium on Microarchitecture*. 2000.
16. Huang, W., et al. Compact thermal modeling for temperature-aware design. *Design Automation Conference*. 2004.
17. Intel® Itanium® Processor Overview, www.intel.com/design/itanium/itanium/
18. 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.
19. 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.
20. Sabry, M.N. Dynamic compact thermal models: An overview of current and potential advances. *International Workshop on Thermal Investigations of ICs and Systems*. 2002.
21. Skadron, K., et al. Temperature-aware microarchitecture. *International Symposium on Computer Architecture*. 2003.
22. Stanford University Compiler Group The SUIF 2 Compiler System, http://suif.stanford.edu/suif/suif2
23. Tsai, C. and S. Kang. Standard cell placement for even on-chip thermal distribution. *International Symposium on Physical Design*. 1999.