Hotspots Elimination and Temperature Flattening in VLSI Circuits

Benjamin Carrion Schafer, *Member, IEEE* and Taewhan Kim, *Member, IEEE*

***Abstract*— This paper proposes a new solution to the problem of eliminating hotspots from gate-level netlits as well as examines the effects of timing constraints on the temperature reduction and the overall temperature flattening on the chip . Our core technique consists of three steps. First, a thermal analysis is carried out for logic netlists. (The netlists are assumed to be either isolated or embedded in a larger system with macro-cells.) We then apply a new technique, called *isothermal logic partitioning technique* (LP-temp), to the netlists, which essentially builds isothermal logic clusters for the netlists and splits each of the logic clusters exceeding the maximum allowed temperature through its hottest point. This will enlarge a contact point for the hotspot to cool down. Finally, the entire system is replaced using a custom designed temperature-aware floorplanner so that the temperature across the entire system is reduced and flattened.**

We have developed a thermal-aware design flow, integrating our thermal-aware logic partitioning technique with a timing and thermal-aware floorplanner. Two cases were analyzed: (tight timing) LP-temp combined with the timing and thermal-aware floorplanner, where the partitioned units by LP-temp are replaced locally considering a tight timing budget (5% timing degradation); (loose timing) LP-temp combined with thermal-aware replacement, considering a loose timing budget (10% timing degradation). From experimentations using a set of benchmark designs, it is confirmed that our temperature reduction technique is effective, generating designs with an average of 5.54% and 9.9% more reduction of peak temperature (on average) for the cases of tight and loose timing than that of the designs by a conventional thermal-aware floorplanner without using LP-temp, respectively.

We also analyzed the effect of our proposed technique on Field Programmable Gate Arrays(FPGAs) in order to contrast its effectiveness on systems with hotspots on hardmacros. Results show that our technique can reduce the temperature in these systems on average 3.40% and 6.61% for the case of loose and tight timing constraints respectively compared to the thermal-aware floorplanner without using LP-temp.

***Index Terms*— Hotspots, Temperature reduction, Temperature flattening, Leakage power.**

I. Introduction

WITH the implacable technology advent, integrated circuits design is facing new challenges. Smaller size transistors allow higher logic densities but involve also the leakage power is now becoming a significant design factor and is reaching a point where it equals the dynamic power consumed in the chip [1]. Low-K dielectrics, triple-oxides, improved design tools and power efficient architectures have avoid so far the most pessimistic forecasts, but extreme power consuming devices are generating great amounts of heat. State of the art microprocessors like the Intel's new Itanium processor now incorporate power controllers on the same die [2] and new FPGAs, like Xilinxs Virtex 5 [3] incorporate on-chip power supply and thermal monitoring capabilities. Reducing temperature increase is becoming a major issue of concern for highly integrated circuit designs that should be addressed in the overall design process in order to keep the chip temperature as low as possible [4].

Temperature has an adverse effect on multiple aspects. It affects the lifetime of the integrated circuit by accelerating the chemical process taking place inside of the chip following Arrhenius equation. Studies show the mean time between failure (MTBF) of an IC is multiplied by a factor 10 for every 30^{o}C rise in the junction temperature [5]. Secondly leakage power is becoming the dominant source of power consumption for new process technologies [1] which grows exponentially with temperature. Moreover, temperature has a negative effect on carrier mobility and therefore switching speed of the transistors and thus the overall timing of the circuit. Specially global signals like the global clock tree suffer increased clock skew [6]. Consequently it is highly desirable to have an even temperature distribution on the chip in order to avoid costly re-design due to timing/temperature and simplify the verification phase. Furthermore, expensive heat dissipaters are required to maintain the chip at a reasonable temperature or could not be used in case of embedded system. Studies have reported that above 30-40 Watts (W), additional power dissipation increases the total cost per chip by more than $1/W [7].

Field-Programmable Gate Arrays (FPGAs) are no exception, especially state of the art FPGAs, like Xilinx's Virtex5 and Altera's Stratix III, based on a 65nm design process and 12 copper interconnect layers. The post fabrication flexibility provided by these devices is implemented using a large number of prefabricated routing tracks and programmable switches. These interconnects can be long, and can consume a significant amount of power. In addition, the programmable switches add capacitance to each wire-segment, which further increasing their power dissipation. Finally, the generic logic structures consume more power than the dedicated logic in ASICs. The

Manuscript received August 29, 2006; revised June 3, 2007. The work was performed while B. Carrion Schafer was a visiting researcher at Seoul National University and has been supported by the Nano IP/SoC Promotion Group of Seoul R&D Program, IT-SoC Program, ETRI project, System IC2010 project of Korea Ministry of Commerce, Industry and Energy, and by the Korea Science and Engineering.Foundation grant funded by the Korea. government(MOST) (No.R01-2007-000-20891-0).

B. Carrion Schafer is with NEC Corporation, EDA R&D Center, Kawasaki, Kanagawa 211-8666 Japan (phone: +81 44 435 9486; fax: +81 44 435 9491; e-mail: schaferb@bq.jp.nec.com).

T.Kim is with Seoul National University, Seoul 151-744, Korea (e-mail: tkim@ssl.snu.ac.kr).

power consumed by the FPGA will lead to heat generation and in turn to the rising of the overall die temperature. This underlines again the importance of embedding temperature reduction techniques in the main design flow.

Temperature is highly dependent on power consumption but depends on a multiple of other factors, making power alone not a valid measure for temperature. Temperature also depends on the placement of the units in the chip. Placing heavy power consuming units close together will intuitively generate an even higher temperature area in the chip as temperature is additive in nature. In contrast, placing power consuming units close to units that have a moderate power consumption will allow the heat generated to dissipate through these units. Other aspects that affect temperature are the execution order of tasks in a unit. Executing tasks one after the other will help the temperature build up whereas spacing the execution of tasks in a unit will allow the unit to have a time to prevent it from heating up. Consequently, temperature should be addressed as an individual design parameter.

Temperature affects many different aspects of the chip and therefore covers multiple research areas. Techniques to improve the heat removal capabilities as well as design of packages and heat sinks have been developed in [8]. At the taks/processor level, runtime thermal management techniques have been developed such as clock gating using real-time temperature sensors [9] or at the compilation level assigning instructions to the coolest available functional unit in VLIW processors [10]. Furthermore, lowering temperature down through power saving techniques such as dynamic voltage scaling (DVS) and sub-banking has been investigated in [11], [12], [13]. On the other hand, in the architectural level, temperature flattening on SoCs by partitioning modules and using the embedded memory as cooling components has been studied in [14]. Resource allocation and binding at high level synthesis stage has been addressed recently in [15] as well as thermal-aware floorplanners in [16], [17].

To the best of our knowledge so far, it has not been attempted to deal with the temperature reduction at the gate level, where a temperature profile of the given netlist is obtained and the temperature inside the given netlist is reduced by recursive partitions of the netlist and re-placements of the resultant units under a timing budget constraint. Working at the gate level has the advantage that extremely accurate power values are obtained as well as exact placement information so that temperature reduction can be accurately tackled. Other approaches at higher levels of abstractions consider a uniform power distribution of each unit, which does not happen in reality (e.g. in a multiplier if small numbers are permanently multiplied the gates corresponding to the lower bits will be switched more often then the gates corresponding to the higher bit). The contributions of this work can be summarized as follows:

- Analyzes gate-level netlists and generates a *thermal map* of the netlists. We allow the designer to specify the granularity of the temperature map. The thermal map will provide a global view of temperature distribution, in addition to the locations of hotspots, coolspots as well as the thermal distribution in the chip.
- Introduces a thermal-aware logic partitioning technique, called *isothermal logic partitioning* (LP-temp), which effectively weakens the hotspots and distributes the temperature evenly for custom logic design as well as for FPGAs.
- Proposes a *thermal-aware design flow* of logic partitioning and placement. A set of comprehensive comparisons between the proposed design flow and the conventional thermal-aware flow is given for custom VLSI design as well as for FPGAs.
- Studies the importance of timing budget on temperature reduction as well as on overall temperature flatting considering designs with gate netlists as well as systems with mixed gate netlists and hardmacros.
- Extends previous work on synthetic benchmarks to incorporate state of the art FPGA hardmacros like embedded multipliers and embedded memories.

This work is made on the assumption that heat flows laterally inside the chip, as shown in multiple previous works, especially thermal-aware floorplanners [18], [19], [20]. The influence of the lateral heat flow will depend on the thermal conductivity of the primary and secondary heat flows of the chip (heat flow through the package and through the pins respectively). In this case we are targeting mostly embedded systems, which have very strict space constraints limiting the type and size of the heatsink. In these cases lateral heat flow becomes extremely important. On the other hand being able to control the lateral heat flow allows also to use a cheaper package (with lower conductivity).

The paper is organized as follows. Section II shows a motivational example to illustrate what the limitations of previous thermal-aware placement method are, and how they can be overcome by logic partitioning. Then, in subsection III.A the thermal simulation method we developed is described followed by details on the procedure of our core technique, isothermal logic partitioning, in subsection III.B. A complete framework combining the isothermal logic partitioning technique with thermal-aware floorplanners is then covered in subsection III.C. Section IV provides a set of experimental results to show the effectiveness of our approach. Finally, section V gives concluding remarks.

II. Motivational Example

When trying to eliminate hotspots or flattening the overall chip temperature, the previous thermal-aware floorplanners assumed that each unit to be placed has an ‘even’ temperature distribution. For example, suppose the floorplan in Fig. 1(a) is the one with the best timing. When we perform a thermal-aware floorplanning without considering logic partitioning, the resultant floorplan will look like the one in Fig. 1(b) where the two cool units Unit3 and Unit4 with temperature of 40°C surround the hot unit Unit1 to take heat from it, conserving the timing constraint. This result occurs because the thermal-aware floorplanner places units so that the heat flow in the chip is to be maximized allowing *hotter* units to cool down faster by placing them close to *cooler* ones. This can be a good way to flatten the overall chip temperature (at a coarse level), but is not a viable way to reduce the appearance of hotspots as a hotspot can happen in the middle of a unit. In this case where a hotspot appears in the center part of a unit, no previously known methods without logic partitioning seem to eliminate its appearance. We can see, in Unit 1 of Fig. 1(b), the appearance of a hotspot because the unit is too big to be completely cooled down by Unit3 and Unit4. On the other hand, Fig. 1(c) shows a resultant floorplan for Units after partitioning it into three pieces and replacement allowing only a small timing degradation. Since a clever partition with re-placement may reduce the possible concentration of heat flow, the hotspot would be small in size compared to that in Fig. 1(a).

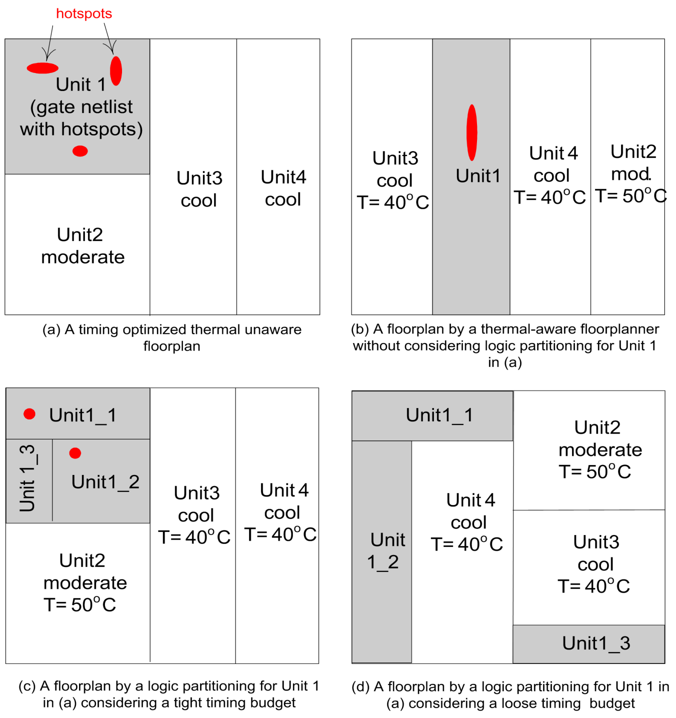

hotspots
Unit 1 (gate netlist with hotspots)
Unit3 cool
Unit4 cool
Unit2 moderate
(a) A timing optimized thermal unaware floorplan
Unit3 cool T= 40°C
Unit1
Unit 4 cool T= 40°C
Unit2 mod. T= 50°C
(b) A floorplan by a thermal-aware floorplanner without considering logic partitioning for Unit 1 in (a)
Unit1_1
Unit 1_3
Unit1_2
Unit3 cool T= 40°C
Unit 4 cool T= 40°C
Unit2 moderate T= 50°C
(c) A floorplan by a logic partitioning for Unit 1 in (a) considering a tight timing budget
Unit1_1
Unit 1_2
Unit 4 cool T= 40°C
Unit2 moderate T= 50°C
Unit3 cool T= 40°C
Unit1_3
(d) A floorplan by a logic partitioning for Unit 1 in (a) considering a loose timing budget

Fig. 1. Motivational example to illustrate the effects of logic partitioing on chip temperature.

Fig. 1(d) show a resultant floorplan which can be obtained by exploiting both of the logic partitioning of Unit1 and a wider replacement, allowing a higher timing degradation. Since in this case, the partitioned units can be relocated in a wider range of 'good' places, the temperature could be further reduced below the limit (i.e., no hotspot as indicated in Fig. 1(d)). The example shown in Fig. 1 illustrates that to weaken the hotspots, logic partition can play an important role, and its effectiveness can be greatly expanded if a thermal-aware replacement and a (thermal-aware) logic partitioning are tightly combined.

Note that the conventional thermal-aware floorplanner with no logic partitioning usually uses a coarse grained (e.g., architectural modules, blocks) power information for every unit in the design. This simplifies the power estimation as well as the thermal simulation of the system. However, if a logic partitioning is taken into account, a more elaborated fine grained thermal estimation technique is required because some parts in a unit might get hotter than others due to the difference of intrinsic switching activity of their transistors.

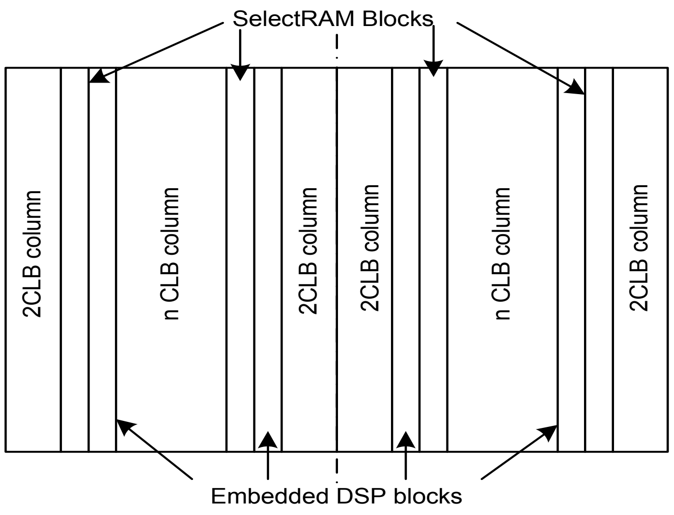


Fig. 2. Simplified Xilinx VirtexII floorplan.

The same technique can be applied to FPGAs. The only aspects to be considered are the embedded hardmacros as shown on Fig. 2 as well a its particular floorplan structure. We will use FPGAs to validate our technique on systems with hardmacros, as shown on the experimental section.

III. Temperature-aware Logic Partitioning and Placement

Our work consists of three parts: (part 1) development of temperature simulator to generate a thermal map, (part 2) generating a thermal map from the data in part 1 and performing a thermal-aware logic partitioning, LP-temp, based on the thermal map, and (part 3) building a complete framework that combines LP-temp with thermal-aware floorplanner. The three parts are described in detail in the following three subsections.

A. Temperature Simulator

In order to have a consistent thermal-aware design flow, a suitable thermal model is needed. On one side it should be accurate enough and on the other side it should be computationally efficient. The thermal model used in this case is based on the known duality between electricity and thermal flow [21] and is based on the model developed by Skadron, *et al.* [22]. Some changes are made from their model as they only consider one type of package (CBGA) with a specific heat sink. In our model, the user can choose a package from a library of different packages so that the equivalent thermal model is generated according to the chosen package. The primary and secondary heat flows will depend on the package type selected. In case of a CBGA package, the primary flow will dissipate heat through the heat sink and the secondary flow will propagate heat through pins of the chip to the PCB.

A thermal mesh is generated on top of the given floorplan, as shown in Fig. 3. The size of each thermal cell is established by the user. A finer mesh will yield a more precise result while taking a longer computational time, whereas a coarser

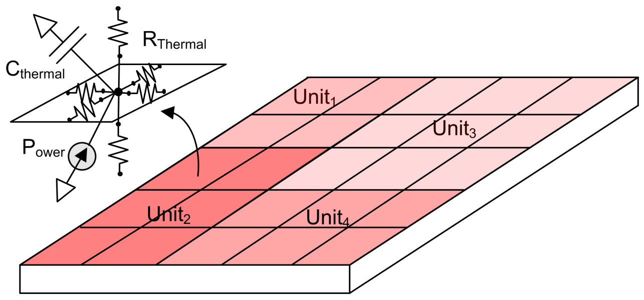


Fig. 3. A conceptual view of thermal equivalent circuit we used.

mesh will provide a less accurate result while being faster as shown on Fig. 4. Each cell consists of 6 resistors, a capacitor and a current source. The thermal capacitor models the transient behavior of the heat flow and the current source of the generated heat. The resistors in the X-Y plane model the 2-D heat flow on the X-Y plane while the resistors in the Z axis model the heat flow through the primary and secondary flows. The thermal resistors are proportional to their length in the direction of the heat transfer and inversely proportional to its heat transfer surface area and thermal conductivity as given by: $R_{thermal} = L/(k \cdot A)$. On the other hand the thermal capacitor is proportional to the area and the thickness: $C = c_p \cdot \rho \cdot L \cdot A$, where c_p is the specific heat and ρ the specific density of the material. The thermal resistance duality allows to solve the heat transfer problem in an analogous manner to electric circuit problems, using the equivalent thermal resistance network, where temperature T is equivalent to the voltage and the heat conduction Q is equivalent to the current. Therefore $\Delta T = Q \cdot R$.

The thermal simulation starts once the equivalent thermal model is generated. A power profile for each unit in the system is passed to the model and the temperature is computed for each thermal cell on each time step. Finite difference equation is used for this propose in order to speed simulation times up. The temperature of each neighboring cell is updated at every time step. The computational time step needs to be small enough so that the heat cannot transfer to the neighboring cell in one time step.

The simulation time is linear with respect to the number of thermal cells as show in Fig. 4, where different ISCAS benchmarks (shown in table I) where thermally analyzed and the running time of each thermal simulation was annotated for different sizes of the grid. It can be seen that the running time of the thermal simulation grows linearly every time the number of thermal cells is doubled.

B. Thermal-aware logic partitioning

Our proposed thermal-aware logic partitioning technique, LP-temp, is performed in two steps: (Step 1) Generating thermal map using the data obtained from the thermal simulation; (Step 2) Finding an ordered list of units to be partitioned and their mesh locations to be cut, and splitting the units.

Step 1: Construction of thermal map: For a given $n \times n$ mesh M of a chip with temperature on each grid, the corresponding

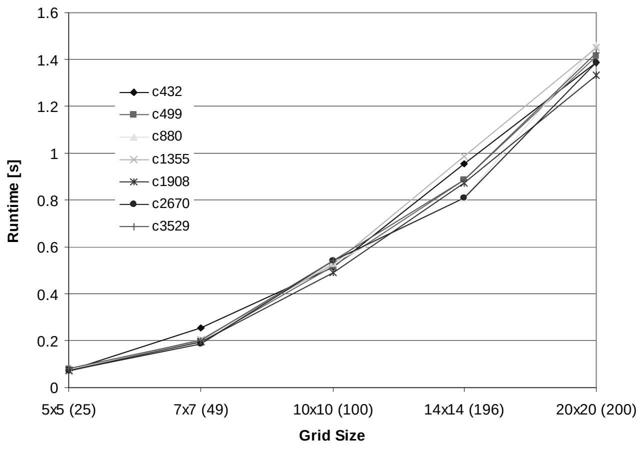


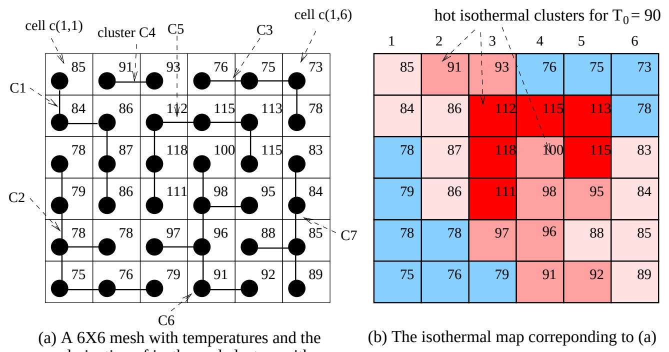
Fig. 4. Runtime of the thermal simulation as a function of the grid size for different ISCAS85 benchmarks.

thermal map is a graphical view of the distribution of temperatures. Let $c_{i,j}$ denote the grid cell of the i^{th} row and j^{th} column of the mesh M, and $t(c_{i,j})$ indicate the temperature on cell $c_{i,j}$. We define terms:

Definition: *Isothermal cluster* of mesh M for temperature t and real value I, called *isothermal interval*, is a subset S of cells in M that satisfies: (i) for each $c_{i,j} \in S$, there is a $c_{k,l} \in S$ such that $c_{i,j}$ and $c_{k,l}$ are adjacent each other in M and (ii) $\lceil \frac{t(c_{i,j})}{I} \rceil = \lceil \frac{t(c_{k,l})}{I} \rceil = \lceil \frac{t}{I} \rceil$. The value of $\lceil \frac{t(c_{i,j})}{I} \rceil$ is referred to as *isothermal level* of $c_{i,j}$ for the isothermal interval I. We represent the isothermal level of $c_{i,j}$ by $iso\_level(c_{i,j}, I)$. For example, in the the grids of Fig. 5(a), $iso\_level(c_{1,1}, 10)$ = $\lceil \frac{85}{10} \rceil$ = 9 and $iso\_level(c_{1,6}, 10)$ = $\lceil \frac{73}{10} \rceil$ = 8.

The construction of thermal map is to find all the sets of isothermal clusters. Each cluster has its isothermal level and multiple clusters may have the same isothermal levels. We can generate the isothermal clusters efficiently by constructing a graph G and extracting all connected components of G: The nodes of G are the cells in mesh M, and there is an edge between two nodes if they are adjacent each other in M and the values of their isothermal levels are identical. From the constructed graph G, we can find all the connected components, each corresponding to a unique isothermal cluster, using a proper graph traversal algorithm (e.g., depth-first search). For example, Fig. 5 shows an example of deriving all isothermal clusters. The dark circled nodes are connected with its neighbor cells that have the same isothermal levels, as shown in Fig. 5(a). Note that there are seven clusters, in which the clusters in three pairs (C1, C7), (C2, C3), (C4, C6), and cluster C5 each has isothermal level of 9, 8, 10, and 11, respectively. The view of isothermal distribution is shown in Fig. 5(b) where we can see the hottest spot is on the cluster C5. We call the clusters which exceed the user specified temperature limit, T_0, *hot isothermal clusters*. For example, in Fig.5(b) if $T_0 = 90^oC$, the set of the hot isothermal clusters is {C4, C5, C6}.

Step 2: Selection of units to be partitioned and cutting points: The candidate units to be partitioned to reduce temperature are the ones that contain at least one cell in the hot



(a) A 6X6 mesh with temperatures and the derivation of isothermal clusters with isothermal interval I = 10

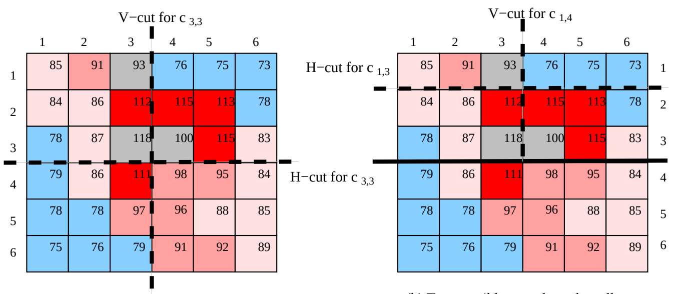
(b) The isothermal map correponding to (a)

Fig. 5. A example showing the derivation of isothermal clusters.

isothermal clusters. The cutting process is as follows:

(1) For each isothermal cluster, we find the cell with the highest temperature among all the cells in the cluster, and sort the cells to their temperatures in nonincreasing order. For the example in Fig. 5(b), the hottest cells are $c_{1,3}$ in C4, $c_{3,3}$ in C5 and $c_{3,4}$ in C6, and the sorted list will be $(c_{3,3}, c_{3,4}, c_{1,3})$.

(2) We iteratively perform partitions, one for each cell in the sorted list. The partition of the logic cells should be done along the boundary of the cell. There are two different types of cut, i.e., vertical cut (V-cut) along the right side of the cell and horizontal-cut (H-cut) along the bottom side of the cell. For example, the dotted two lines in Fig. 6(a) show the two cut lines for cell $c_{3,3}$. We choose the cut line which cuts the fewer number of interconnects. Then, we delete all the cells in the sorted list whose isothermal clusters are also cut by the chosen cut line. Let us assume that H-cut in Fig. 6(a) cuts fewer interconnects than V-cut. Then, we can see that cell $c_{3,4}$ will also be removed from the consideration of cutting point in the list as its cluster C6 has also been cut by the H-cut chosen for $c_{3,3}$. Fig. 6(b) shows the two cuts for $c_{1,3}$. The cut selection procedure is then repeated for $c_{1,3}$.



(a) Two possible cuts along the cell c $_{3,3}$

(b) Two possible cuts along the cell c $_{1,4}$ after the selection of the cut for c $_{3,3}$

Fig. 6. A example showing the logic partitioning steps based on the isothermal clusters.

Fig. 7 summarizes the procedure of our thermal-aware logic partitioner LP-temp. The most time consuming part in LP-temp is the *while*-loop, which is bounded by $O(n^2e)$ where e is the number of nets in the circuit because the loop take at most n^2 times since $|L| = O(n^2)$, where n^2 is the number of cells in M, and counting cuts for H-cut and V-cut takes $O(e)$ time. However, note that since $|L|$ is practically a small value, much less than n^2, we can assume the practical time complexity of LP-temp to be $O(ke)$ where k is a certain constant.

```
LP-temp: Thermal-aware logic partitioner($\mathcal{F}$,$M$,$T_0$,$I$)
/* $\mathcal{F}$: input floorplan of logic cells
   $M$: $n \times n$ mesh, $I$: temperature interval
   $T_0$ : upper limit constraint of temperature */
/* Part 1 */
• Apply thermal simulation to $\mathcal{F}$ with mesh $M$ and the
  duality relation between electricity and thermal flow;
/* Step 1 of Part 2 */
• Generate thermal map on $M$;
/* Step 2 of Part 2 */
• Generate thermal clusters;
• Extract a list $L$ of hottest cells of clusters exceeding $T_0$;
while ($L \neq \emptyset$) do /* partition the clusters */
    • Remove the hottest cell $c$ from $L$;
    • Select the cut, between H-cut and V-cut, with
      fewer wire cuts and apply the cut to $\mathcal{F}$;
    • Remove the cells in $L$ whose clusters were
      also partitioned by the cut;
endwhile;
return $\mathcal{F}$;
```

Fig. 7. A summary of the procedure of our logic partitioner, LP-temp, for eliminating hotspots.

C. Integration of thermal-aware logic partitioner and floorplaner

The partitioned result produced by LP-temp is then used as an input to a thermal-aware floorplanner to find a better placement for the units. The initial placement of the logic gate netlist is timing optimized. Therefore every split and replacement will degrade the netlist timing. The maximum timing interval degradation allowed needs to be specified so that only the valid new placements are accepted. It looks obvious that as presented in the motivational example, the more generous this interval is further apparat units can be placed and therefore our technique will yield better results. Fig. 8 shows the flow graph of our entire framework. As indicated in the loop of the flow of the system in Fig. 8, once a new floorplan result is obtained, LP-temp is again applied to the units in the floorplan. The process then repeats until there is no hotspot any more or there is no more reduction on the highest temperature.

The heat generated by the hotspots will spread following the path of largest gradient and will increase the temperature of neighboring units. *Hot* units will need to be placed close to *cooler* units in order to diffuse the temperature evenly across the die, reducing the highest temperature of the circuit. In our work, we developed a slicing floorplanner as it has been shown that the hierarchical nature of slicing structures entails many algorithmic advantages over non-slicing ones [23]. They are much easier to handle, and reduce data structure complexity and computational time. Our floorplanner is based on Wong's slicing algorithm in [24]. The floorplanner is hierarchical

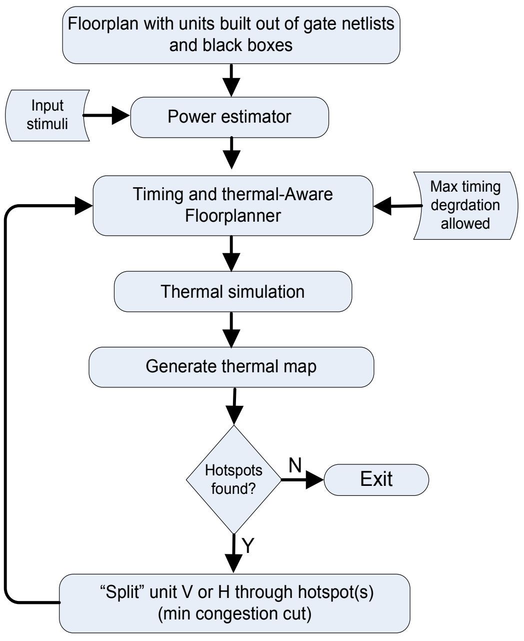

Floorplan with units built out of gate netlists and black boxes
Input stimuli
Power estimator
Timing and thermal-Aware Floorplanner
Max timing degrdation allowed
Thermal simulation
Generate thermal map
Hotspots found?
N
Exit
Y
"Split" unit V or H through hotspot(s) (min congestion cut)

Fig. 8. The entire flow of our proposed framework.

allowing a top unit to be built of multiple units and/or a gate-level netlist.

The cost function of the simulated annealing floorplanner is given by COST = $\alpha A + \beta W + \gamma T$. The weighting factors α, β, γ can be modified in order to represent the importance of minimizing the total area A), wire length(W), or maximal temperature(T). In order to place *cooler* units close to *hotter* ones, the cost function was modified in order to maximize the heat diffusion between two adjacent blocks (D). Specifically, heat diffusion is proportional to the temperature difference and the length of their contact area [25]. The cost function redefined is COST = $\alpha A + \beta W - \gamma D$, in order to consider the heat diffusion as one of its parameters. D has a negative sign as we want to maximize the heat diffusion. In our case as we are analyzing the effects of the floorplan on temperature we chose the thermal diffusion coefficient (γ) twice as large as the area (α) and wire length (β) one.

IV. Experimental Results

First, we describe the experimental setup for the generation of initial floorplans for custom logic designs of the conventional and our proposed thermal-aware design flow here investigated. Then, we show a set of comprehensive results obtained, together with explanations on the implication and analysis of the data. Secondly we apply our methodology to FPGAs showing how our temperature reduction and flattening technique can also be effective on FPGAs with multiple hotspots (some on the logic and some on hardmacros).

A. Experimental Setup for Custom Logic

To test our temperature reduction flow, we have developed our own integrated environment in C++. This tool (that we call hotkiller) has a main program with multiple external subroutines (libraries) that are called upon needed. The libraries correspond to the major steps in the flow, which are the power estimator, thermal-aware floorplanner, thermal simulator, thermal-aware partitioner and a synthetic benchmark generator (used for the FPGA benchmarks) as shown on Fig 9. The tool has a set of libraries as inputs so that either a custom floorplan can be specified or one from the libraries can be chosen. Libraries include a set of Xilinx FPGAs. The output of the design flow will be a temperature optimized design.

TABLE I
TYPES AND SIZES OF TESTED BENCHMARKS BASED ON THE ISCAS85 BENCHMARK CIRCUITS

Benchmark	Netlists (gates)	# Total Gates
Bench 1	c432, c499, c880, c1355 (160,202,383,546)	1291
Bench 2	c499, c880, c1355, c1908 (202,383,546, 880)	2011
Bench 3	c880, c1355, c1908, c2670 (383,546, 880,1269)	3078
Bench 4	c880, c1908, c2670, c3529 (383,880,1269,1669)	4201
Bench 5	c1355, c1908, c2670, c3529 (546,880, 1269, 1669)	4364
Bench 6	c1355, c1908, c2670, c5315 (546,880, 1269,2307)	5002
Bench 7	c1908, c2670, c3529, c5315 (880, 1269, 1669,2307)	6125
Bench 8	c1355, c2670, c3529, c6288 (546, 1269, 1669, 2416)	5900
Bench 9	c1355, c2670, c5315, c7522 (546, 1269, 1669,3512)	7634
Bench 10	2670, c5315, c6288, c7522 (1269, 1669, 2416,3512)	9504

To test the effectiveness of our proposed design flow integrated with logic partitioner LP-temp, we generate a random floorplan with 4 different gate netlists assigned to each unit in the floorplan for all cases as shown in Fig.10. The tested circuits are taken from the ISCAS85 benchmarks and have a total combined size that range between 1291 to 9505 gates, as

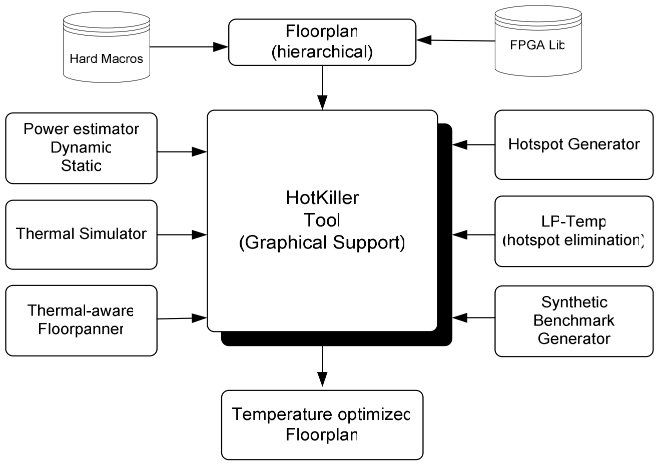

Hard Macros
Floorplan (hierarchical)
FPGA Lib
Power estimator Dynamic Static
Thermal Simulator
Thermal-aware Floorpanner
HotKiller Tool (Graphical Support)
Hotspot Generator
LP-Temp (hotspot elimination)
Synthetic Benchmark Generator
Temperature optimized Floorplan

Fig. 9. Hotkiller tool block diagram.

TABLE II
A COMPARISON OF PEAK TEMPERATURES PRODUCED BY THE CONVENTIONAL THERMAL-AWARE FLOORPLANNER AND OUR THERMAL-AWARE LOGIC PARTITIONER.

		Thermal-aware floorplan only			LP-temp + 5% timing floorplan				LP-temp + 10% timing floorplan			
	T_{init}	T_{peak}	ΔT[^{o}C]	ΔT[%]	T_{peak}	ΔT[^{o}C]	ΔT[%]	#	T_{peak}	ΔT[^{o}C]	ΔT[%]	#
Bench 1	64.71	60.32	4.39	6.78	58.54	6.17	9.53	10	55.67	9.04	13.97	8
Bench 2	66.38	62.22	4.16	6.27	59.68	6.7	10.09	8	56.64	9.74	14.67	8
Bench 3	64.98	59.23	5.75	8.85	57.31	7.67	11.80	9	55.91	9.07	13.96	8
Bench 4	65.65	62.56	3.09	4.71	57.91	7.74	11.79	8	55.81	9.84	14.99	8
Bench 5	66.89	62.57	4.32	6.46	60.31	6.58	9.84	9	57.52	9.37	14.01	9
Bench 6	67.23	63.84	3.39	5.04	61.75	5.48	8.15	8	57.62	9.61	14.29	10
Bench 7	67.89	65.64	2.25	3.31	59.94	7.95	11.71	9	59.19	8.7	12.81	9
Bench 8	69.41	66.37	3.04	4.38	61.84	7.57	10.91	10	59.74	9.67	13.93	11
Bench 9	71.23	69.01	2.22	3.12	63.31	7.92	11.12	11	62.4	8.83	12.40	12
Bench 10	70.81	69.11	1.7	2.40	62.46	8.35	11.79	12	59.97	10.84	15.31	12
Avg	67.51	64.09	3.43	**5.13**	60.31	7.21	**10.67**	9	58.05	9.47	**14.03**	9

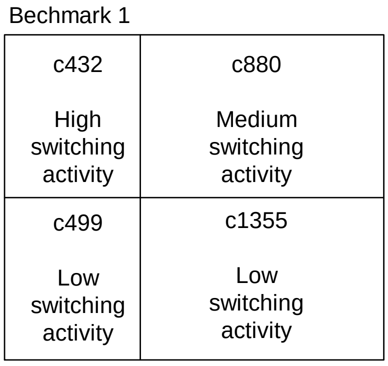


Fig. 10. Benchmark 1 floorplan example

shown in table I. An input stimuli file with different switching activities is generated with 10,000 input values and assigned to each gatenetlist. Then, the switching activity is simulated using the embedded power estimator computing the capacitances for each gate based on [26]. To achieve a thermal gradient in the chip different netlists are assigned different types of stimuli files in order to have one with a high switching activity, one with a medium switching activity and the last two with low switching activity as shown on Fig.10, where a 4 unit floorplan is shown with 4 different netlist assigned to each unit each with different switching activity.

We then generated three different solutions by applying the following three approaches. The first one is the thermal-aware floorplanning without logic partitioning (thermal-aware floorplan only in Table II). This solution will place the units thermally as convenient as possible, placing *cooler* units near to *hotter* ones. The second as well as the third solutions use our logic partitioner LP-temp, allowing different levels of timing degradation (5% and 10%).

B. Experimental Results for Custom Logic

Table II shows a comparison of peak temperatures used by the conventional floorplanner, our logic partitioner with replacement allowing up to 5% and 10% of timing degradation. T_{init} in the second column of the table represents the maximum temperature at the initial floorplan of the corresponding circuits without logic partition. On the other hand, T_{peak} at the 3^{rd}, 6^{th}, and 10^{th} columns indicates the final peak temperature used by the corresponding three design flows, respectively. For thermal-aware floorplans + thermal-aware logic partition , the total number of units produced by LP-temp is also recorded in the columns marked with '#'. $\Delta T[^oC]$ and $\Delta T[\%]$ columns represent the amounts of the peak temperature reductions in $[^oC]$ and $[\%]$ by applying the corresponding design flows to the initial floorplans of circuits. From the the table, we can see that thermal-aware floorplan only achieves in some cases a good temperature reductions. For example, for small circuits (e.g. Bench 1 to Bench 3) the thermal-aware floorplanner was able to reduce the maximum temperature between 6.27% to 8.85%. For larger circuits (Bench 4 to Bench 10) it could only reduce the temperature between of 2.4% to 6.46%. The worst results are obtained for large sized circuits where the hotspot is located almost at the center of the unit. In this case, the floorplanner takes hard time to reduce the hotspot temperature as the *cooler* units will have no influence on it (e.g. Bench 9 and Bench 10), while our technique consistently reduces the temperature of any benchmark independently of its size by an average of 10.67% and 14.03% for the 5% and 10% timing degradation respectively.

Fig. 11 shows how the peak temperature is reduced at every iteration for both of our thermal-aware design flows, compared with the conventional flow, i.e., thermal-aware floorplanner only. It can be seen that the temperature is reduced until the 3^{rd} iteration. At this point, no further cuts were performed as the temperature reduction would be minimal, which means at the point, our design flows would not yield any further significant temperature reduction.

Fig. 12 shows how the leakage power is reduced at each benchmark for the different experimental setups as well as the average leakage power savings, considering a 65nm process technology, according to the calculations in [26]. It clearly shows the impact of temperature on leakage power, as leakage power grows exponentially with temperature.

Figs. 14 and 13 also shows how other metrics (like total wirelength and maximum delay) behave for the different tech-

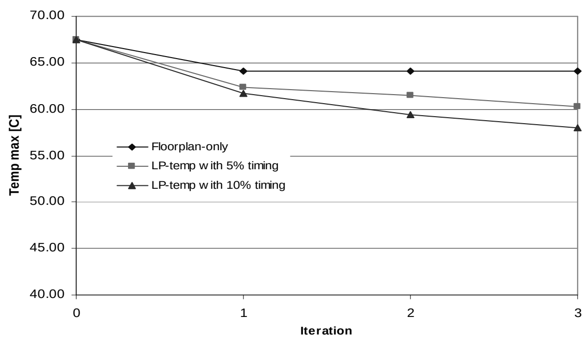


Fig. 11. Average temperature reduction by each of the temperature reduction approaches after each iteration.

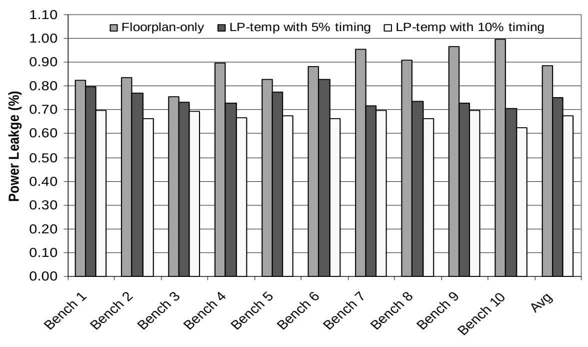


Fig. 12. Average Leakage powers for all three hotspot types by each of the temperature reduction approaches.

niques. Fig. 13 shows the maximum delay for each benchmark. Though the maximum allowed timing variation is 5% and 10% the average timing penalty is between 3.83% for the case were up to 5% of timing degradation was allowed and 8.12% for the 10% case. A timing degradation of 10% was allowed to thethermal-aware floorplan.

Fig. 14 shows the total maximum wirelength of each benchmark. It can be noticed that the average wirelength increases for the looser timing budget (10% timing) as units are now allowed to be placed further away.

In terms of run time, our floorplanner used in each of the three design flows dominates the run time since it has to perform a thermal simulation in each new annealing step. The isothermal clustering and logic partitioning take around 3% of the total run time (ran on a Pentium IV at 3.0 GHz with

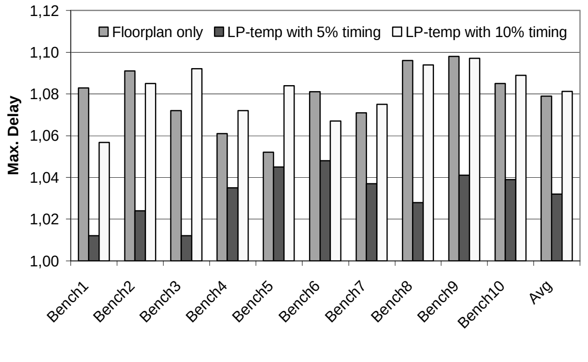


Fig. 13. Maximum critical path delay by each of the temperature reduction approaches.

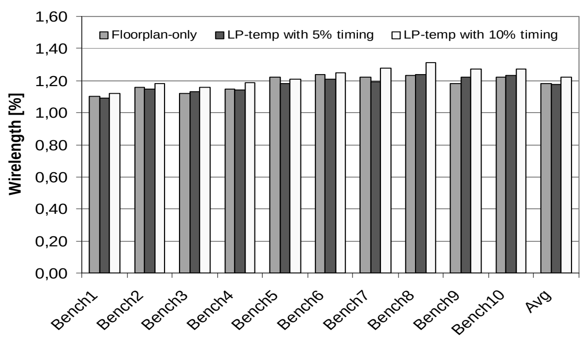


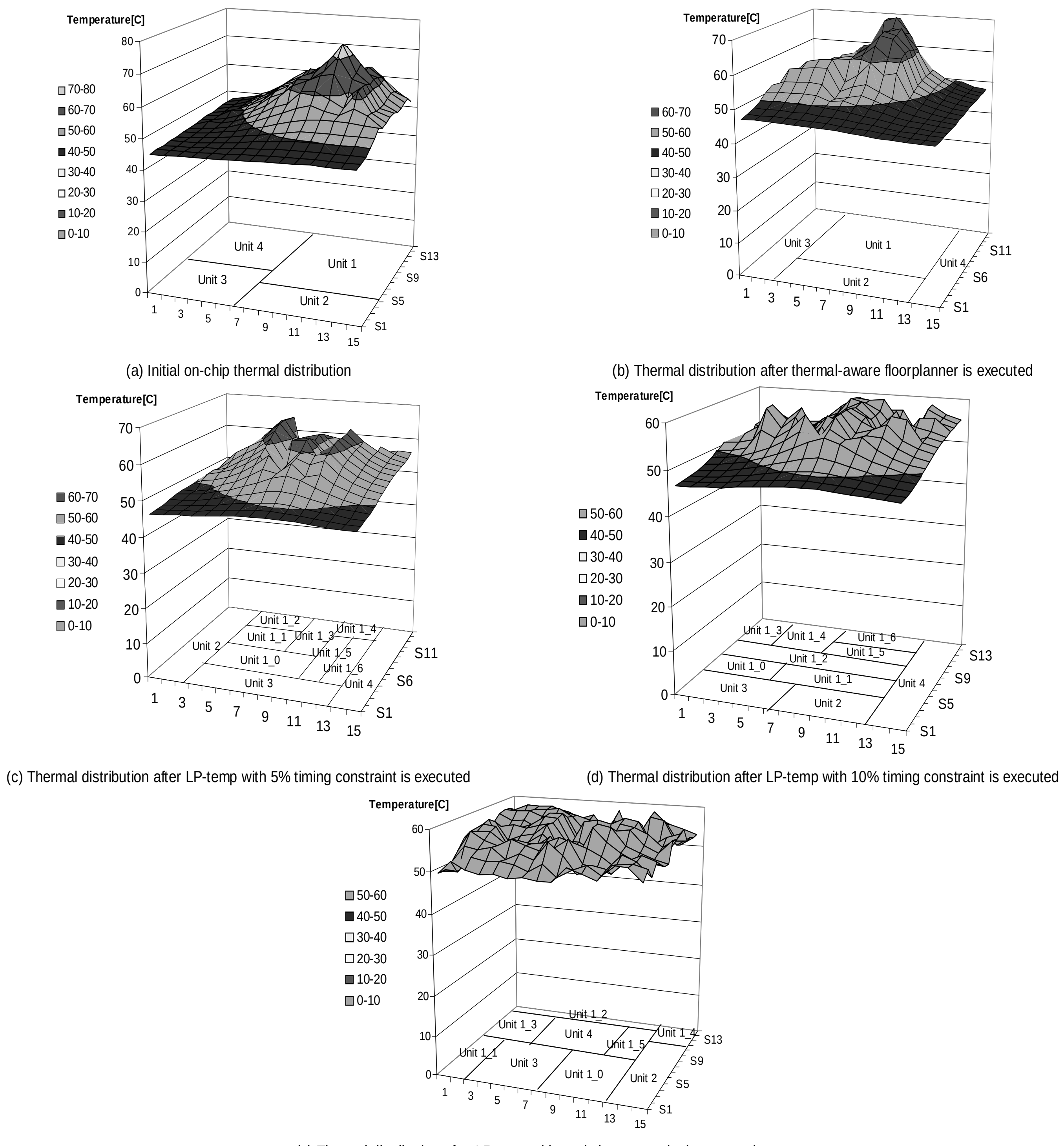
Fig. 14. Total Wirelength for all three hotspot types by each of the temperature reduction approaches.

1 GByte of RAM) Run time rises for larger benchmarks as the thermal simulation also takes longer as well as with the number of newly generated units after each split (#)..

Finally Fig. 15 shows the thermal distribution on the die for Bench 4 for the different techniques. Fig. 15(a)shows the initial thermal map of the chip and its floorplan on the x-y plane. The temperature is extremely low on units 2-4 as their switching activity is relatively low. In contrast Unit 1 has a peak at temperature 65.65^oC. Fig. 15(b) shows the thermal distribution after the thermal-aware floorplanner has been ran. On the x-y plane it can be noticed how Unit 1 is now partially surrounded by the *colder* units allowing it to dissipate part of its heat through them. The peak temperature is reduced to 62.22^o but the thermal gradient on the chip is still noticeable. The rest of the cases, show the thermal distribution on the chip after applying LP-temp using the different timing constraints. Fig. 15(c) shows the 5% increase from the maximum delay, Fig. 15(d) up to 10%, while Fig. 15(e) does not consider timing at all. It can be clearly seen how the temperature is reduced the most for the latter case as well as the temperature is flattened the most across the chip. Not considering any timing constraints will allow each unit to be placed anywhere on the chip resulting in a higher temperature reduction and furthermore total lower temperature gradient. In the case of only 5% timing, Fig. 15(c) clearly shows some isolated temperature peaks very close to each other, indicating that the single generated hotspot is split in multiple smaller ones reducing therefore the peak temperature. On the other hand a the loose timing constraints imposed in Fig. 15(d) allow the partitioned units to be re-placed further away of each other reducing the peak temperature further, placing cooler units in between hotter ones reducing at the same time the temperature gradient in the chip, evening the temperatures even further.

C. Experimental Setup for FPGAs

In this section we validate our temperature reduction and flattening technique on state of the art FPGAs with hard-macros. In order to perform the tests on FPGAs suitable benchmarks were needed. One option would have been to take the ISCAS benchmarks and map these to an FPGA using [28]. This would ensure the use of some real benchmarks, but has the drawback that these benchmarks were not designed for FPGAs and therefore do not reflect real needs in FPGA designs



(a) Initial on-chip thermal distribution

(b) Thermal distribution after thermal-aware floorplanner is executed

(c) Thermal distribution after LP-temp with 5% timing constraint is executed

(d) Thermal distribution after LP-temp with 10% timing constraint is executed

(e) Thermal distribution after LP-temp with no timing constraint is executed

Fig. 15. On-chip thermal distributions for the techniques for benchmark4.

and do not consider their hardmacros, which is one of the aspects we want to investigate in this section. For instance the ISCAS85 benchmark circuits were developed specifically for the evaluation of ATPG (Automatic Test Pattern Generation) tools.

We therefore decided to use synthetic benchmarks for our experimental results. These present multiple advantages over real benchmarks. First of all as many benchmarks as needed can be generated automatically. Secondly, they provide full control over the benchmark's most important characteristic parameters, such as circuit size, interconnection structure and functionality. The main advantage is the controllability of a single characteristic parameter at a time. The major drawback of synthetic benchmarks is that it is hard to prove that they are equivalent to certain real benchmarks.

TABLE III
INFORMATION OF THE GENERATED SYNTHETIC BENCHMARKS

Benchmark	LUTS	Mults	Mem	Rents	Net degree
syn_500	500	0	0	0.6	2.5
syn_1000	1000	0	0	0.6	2.5
syn_1500	1500	2	0	0.6	2.5
syn_2000	2000	2	0	0.6	2.5
syn_2500	2500	4	2	0.6	2.5
syn_3000	3000	4	2	0.6	2.5

TABLE IV
TYPES AND SIZES OF FPGA BENCHMARK CIRCUITS

Benchmark	Netlists (LUTs)	# Total LUTs
Bench 1	syn_500, syn_1000, syn_1500, syn_2000	5000
Bench 2	syn_500, syn_1500, syn_2000 , syn_2500	6500
Bench 3	syn_500 , syn_1500 , syn_2000 , syn_3000	7000
Bench 4	syn_1000 , syn_1500, syn_2500, syn_3000	8000
Bench 5	syn_1000 , syn_2000, syn_2500, syn_3000	8500
Bench 6	syn_1500, syn_2000, syn_2500, syn_3000	9000

Two parameters are extremely important to obtain realistic benchmarks: (a) The Rent's exponent and (b) the net degree distribution as explained in detail in [27]. We extended the previous work on synthetic benchmarks to adapt it to modern FPGAs with embedded hardmacros like embedded hardware multipliers and memory, being able to generate benchmarks of a given amount of LUTs, number of embedded memory blocks, number of hardware multipliers as well as the Rent's exponent and net degree distribution of the benchmark.

Table IV-C shows the different benchmarks generated to test the temperature reduction technique on FPGAs. 6 benchmarks were generated in total with different size ranging from 500 to 3000 LUTs. As we are interested in checking our temperature reduction technique on systems with hardmacros the syn_1500 to syn_3000 consist of a logic netlist with embedded multipliers ranging from 2 to 4 as well as embedded memory blocks for syn_2500 and syn_3000.

The Rent's parameter (R) is a measure of the interconnection complexity of a logic circuit. It has been shown that R normally ranges between 0.45 and 0.75, where the lowest value correspond to extremely regular circuits like memories and the highest values to custom logic circuits of complex circuits. We therefore decided to make R or all of the benchmarks 0.6. On the other hand it was observed in [29] that more than 75% of net in real circuits are 2-3 terminal nets. This will have a big influence on the resultant circuit especially at current technologies where interconnect is becoming a dominant factor in terms of delays and power consumption. We therefore decided to choose a net degree factor of 2.5. The generated benchmarks were mapped on Xilinx Virtex II XC2V1000 FPGA, which has 10,240 LUTs, 40 embedded multipliers as well as 40 embedded memory blocks [3].

In order to achieve a thermal gradient 6 benchmarks were generated using the previously described synthetic benchmarks, each composed of 4 individual netlists, each with a different switching activity associated to it, as explained at the gatenetlist previous section. Table IV show the different configurations of the benchmarks used.

The benchmarks were initially placed and routed on the selected FPGA optimized for timing.

D. Experimental Results for FPGAs

Table V shows a comparison of peak temperatures used by the three different approaches explained in the previous section. It can be seen that for benchmarks Bench 1,2,5 and 6 our proposed technique LP-temp performs as expected reducing the peak temperature in 8.90-10.98% for the tight timing variant and 11.48-14.50% in the loose timing case. In benchmarks Bench 3 and 4 our proposed technique does not perform as well reducing the peak temperature less then the rest of the benchmarks in each case due to the fact that a hotspot is located on a hardmacro (i.e. embedded multipliers). Our technique does only apply for logic netlists that can be partitioned and re-placed. In the case where hardmacros are hotspots our technique can reduce the temperature by partitioning the hotspots close to the hardmacros with hotspots and placing these as far away as possible as well as separating the hardmacros as far away as possible and surround them as well as much as possible with cooler units. This explains why the peak temperature is only reduced in the case of hardmacros being hotspots by 6.35-7.53% in the tight timing constraint case and 9.80-10.68% in the loose timing case.

Figs. 16, 17, 18 show how different metrics (leakage power, maximum delay and total wirelength) change for each benchmark. Leakage power is reduced between 5.11% using the thermal-aware floorplanner to 7.32% using the loose timing LP-temp.

Fig. 19 shows the thermal map of the FPGA in 5 different scenarios for benchmark Bench 3. Fig. 19(a)shows the initial temperature map before any temperature reduction technique is applied. 4 hotspots can be clearly identified. 2 in the FPGAs logic and 2 hotspots on the hardmacros. Fig. 19(b) shows the thermal map after the thermal-aware floorplanner is ran. It can be noticed how Unit 3 has been rotated so that both hotspots are not placed too close lowering the overall peak temperature. Fig. 19(c) presents the thermal map after LP-temp is applied using a tight timing constraint (5%). Same as in the logic gatenetlis case in Fig. 15(c) the hotspots corresponding to the logic gatenetlist are weakened due to the partitioning but can not be re-placed too far away due to the tight timing constraints. As temperature is additive in nature the hardmacros' peak temperature is therefore also reduced slightly. Fig.15(d) shows the thermal map of the FPGA after LP-temp is applied using a loose timing constraint (10%). Temperature in the logic gatenetlists hotspots is futher reduced as well as the temperature in the hardmacros as the hotter units are re-placed further away from them. The last figure (Fig.15(e))shows the final thermal map when not timing constraints are used. the temperature is very even among most of the FPGA except of the 2 hardmacros with the hotspots where the temperature can only be reduced applying our technique up to its self generating temperature, due to its intrinsic power consumption.

V. Conclusion

Temperature on a chip is increasingly becoming a critical design consideration of integrated circuits. Especially, the hotspots cause devastating effects on leakage power, circuit delay, and circuit reliability. In this work, we proposed an effective hotspot elimination technique by introducing a concept called *thermal-aware logic partitioning* (LP-temp). By combining LP-temp with a timing and thermal-aware floorplanner, it was shown that the hotspot temperatures in circuits were reduced by up to 14.99^{o}C. LP-temp affords high flexibility in that in some case, it can be applied to the finest logic granularity, and in other case, it can also be applied to the (higher) level of hardware building blocks that can be subsequently partitioned into smaller units. Compared to the custom thermal-aware floorplanner our design flow was able to further reduce the peak temperature by 5.54% and 9.9%, with 5% and 10% timing degradation respectively and subsequently save up to 37.24% total leakage power over that by the thermal-aware floorplanner without LP-temp.

We also presented a study about the influence of timing constraints in the peak temperature reduction as well as on the on chip thermal gradient, showing that looser timing constraints combined with LP-temp can reduce the temperature further and will flatten the temperature distribution.

In the last part of this paper we presented the behavior of our temperature reduction techniques in integrated circuits with fixed hardmacros (like FPGAs). We noted that the our technique can still reduced the overall temperature though not as significant as in the custom logic case as our technique only applies to gate netlists.

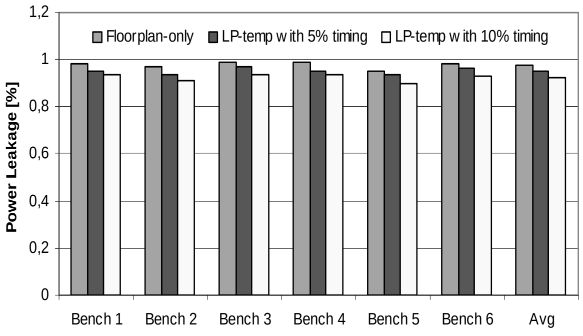


Fig. 16. Leakage power for the given FPGA benchmarks

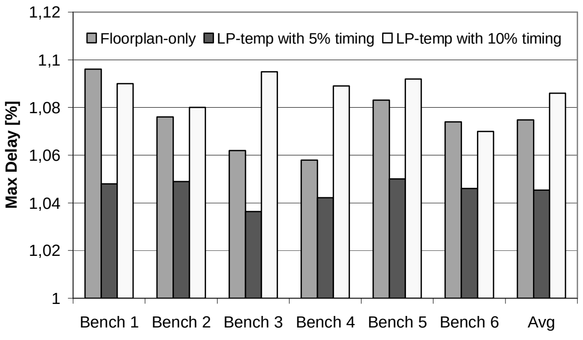


Fig. 17. Delay for the given FPGA benchmarks

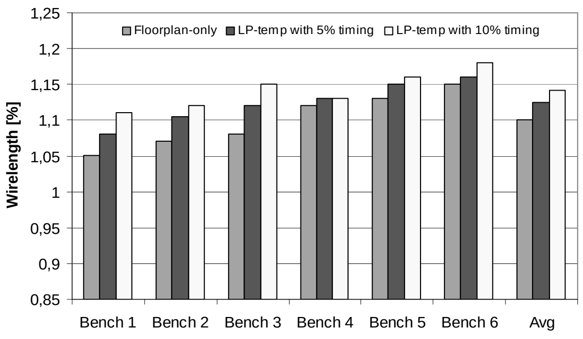
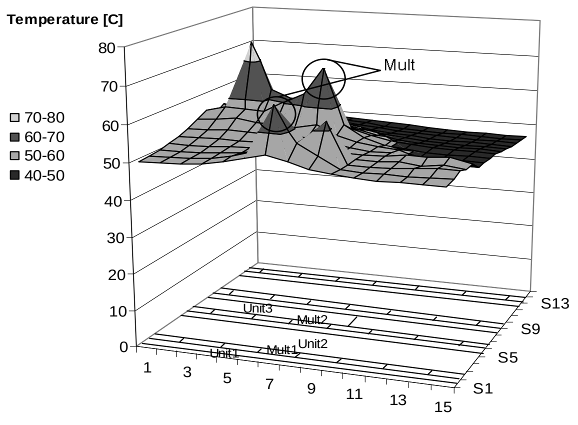


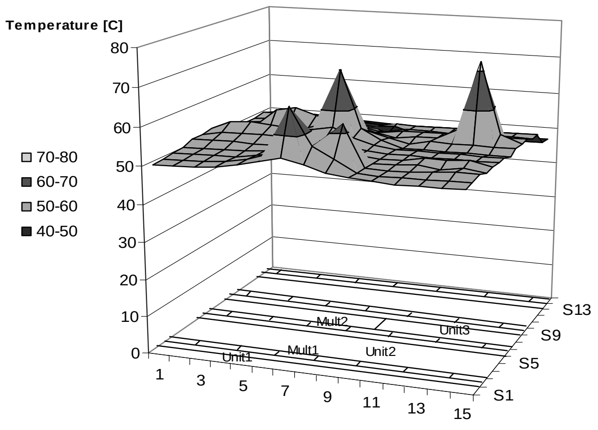
Fig. 18. Wirelength for the given FPGA benchmarks

References

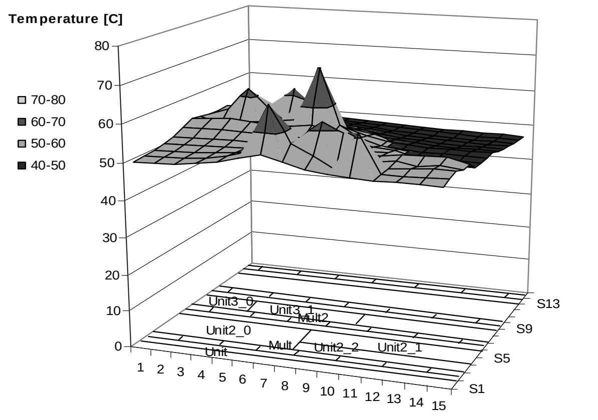
[1] F. Fallah and M. Pedram, "Standby and Active Leakage Current Control and Minimization CMOS VLSI Circuits," *IEICE Transactions on Electronics*, Vol. 88, pp. 509-519, 2005.
[2] T. Fischer, J Desai, B. Doyle. S. Naffziger and B. Patella, "A 90-nm Variable Frequency Clock System for a Power-Managed Itanium Architecture Processor," *IEE Journal of Solid-State Circuits*, Vol.41, No.1, January 2006.
[3] Xilinx, "Xilinx datasheet," *wwww.xilinx.com.*
[4] K. Banerjee and A. Mehrotra, "Global (Interconnect) Warming," *Circuit & Devices*, Vol. 17, pp. 16-32, September 2001.
[5] National Semiconductor, USA, *Understanding Integrated Circuit Package Power Capabilities,* www.national.com, April 2000.
[6] A.H. Ajami, M. Pedram and K. Banerjee, "Effects of none-uniform substrate temperature on the clock signal integrity in high performance designs," *Proc. CICC*,pp. 223-236, 2001.
[7] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, Jul-Aug. 1999.
[8] M.N. Sabry, "Dynamic Compact Thermal Models: An Overview of Current and Potential Advances" *International Workshop on Thermal Investigations of ICs and Systems*, pp.1-18, 2001.
[9] S. Gunther, "Managing the Impact of Increasing Microprocessor Power Consumption" *Intel Technology Technology Journal*, 2001.
[10] B. Carrion Schafer, Y. Lee andTaewhan Kim, "Temperature-aware Compilation for VLIW Processors," *Real-Time Computing Systems and Applications (RTCSA)*, pp. 426-431, September 2007.
[11] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High Performance Microprocessors, " *International Symposium on High-Performance Computer Architecture*, pp. 171-182, 2001.
[12] L. Cao, J.P. Krusius, M.A. Korhonen, T.S Fisher, "Transient Thermal Management of Portable Electronics using Heat Storage and Dynamic Power Dissipation Control," *IEEE Transactions on Components, Packaging, and Manufacturing Technology*, Vol. 21, No. 1, Part A, pp. 113-123, 1998.
[13] M. Huang, J. Renau, S.M. Yoon and J. Torrellas, "A Framework for Dynamic Energy Efficiency and Temparature Management," *International Symposium on Microarchitecture*, pp.202-213, 2000.
[14] T. Sato, J. Ichimiya, N. Ono, K. Hachiya and M. Hashimoto, "On-Chip Thermal Gradient Analysis and Temperature Flattening for Soc Design," *IEICE Trans. Fundamentals*, Vol. E88-A No.12, pp. 1074-1077, December, 2005.
[15] R. Mukjerjee, S. O. Memik, and G. Memik, "Temperature-Aware Resource Allocation and Binding in High-Level Synthesis," *Design Automation Conference (DAC)*, pp. 196-201, 2005.
[16] C. H. Tsai and S. M. Kang, "Standard Cell Placement for Even On-Chip Thermal Distribution," *International Symposium on Physical Design*, pp. 179-184, 1999.
[17] C. C. Chu and D. F. Wong, "A Matrix Synthesis Approach to Thermal Placement," *IEEE Transactions on Computer-Aided Design*, Vol. 17, No. 11, pp. 163-168, November 1998.
[18] C.H. Tsai and S.M. Kang,"Standard Cell Placement for Even On-Chip



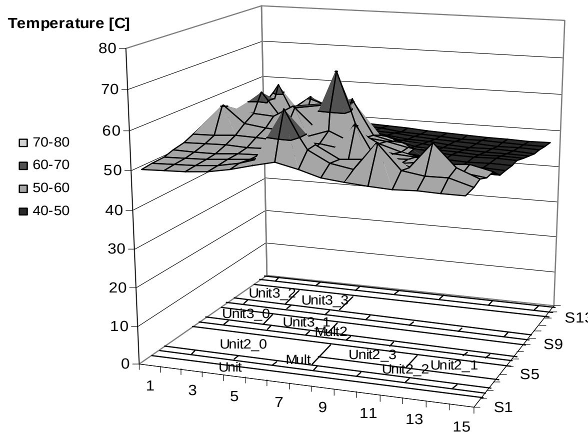
(a) Initial on-chip thermal distribution in the FPGA



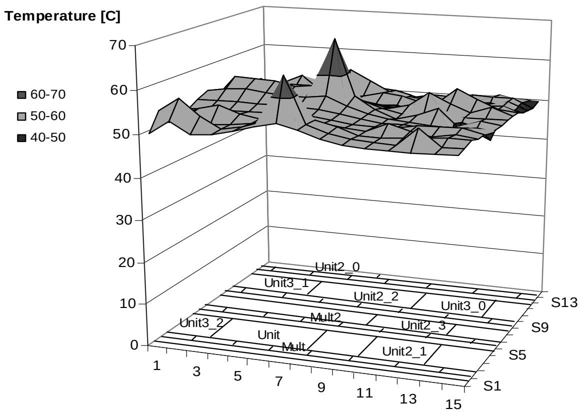
(b) Thermal distribution after thermal-aware floorplanner is executed in the FPGA



(c) Thermal distribution after LP-temp with 5% timing constraint is executed in the FPGA



(d) Thermal distribution after LP-temp with 10% timing constraint is executed in the FPGA



(e) Thermal distribution after LP-temp with no timing constraint is executed in the FPGA

Fig. 19. On-chip thermal distributions for the techniques for benchmark 3 (Bench 3).

TABLE V

A COMPARISON OF PEAK TEMPERATURES ON XILINX'S XC2V1000 FPGA PRODUCED BY THE CONVENTIONAL THERMAL-AWARE FLOORPLANNER AND OUR THERMAL-AWARE LOGIC PARTITIONER COMBINED WITH THE THERMAL-AWARE FLOORPLANNER FOR THE CIRCUITS IN TABLE IV.

		Thermal-aware floorplan only			Thermal-aware LP + 5% timing floorplan				Thermal-aware LP + 10% timing floorplan			
	T_{init}	T_{peak}	ΔT[^{o}C]	ΔT[%]	T_{peak}	ΔT[^{o}C]	ΔT[%]	#	T_{peak}	ΔT[^{o}C]	ΔT[%]	#
Bench 1	68,53	64,76	3,77	5,50	61,81	6,72	9,81	50	60,66	7,87	11,48	41
Bench 2	68,33	63,71	4,62	6,76	60,83	7,50	10,98	47	58,54	9,79	14,33	47
Bench 3	70,07	67,23	2,84	4,05	65,62	4,45	6,35	36	62,59	7,48	10,68	42
Bench 4	68,88	66,28	2,60	3,77	63,69	5,19	7,53	41	62,13	6,75	9,80	44
Bench 5	67,40	61,58	5,82	8,64	60,76	6,64	9,85	47	57,63	9,77	14,50	48
Bench 6	60,78	58,23	2,55	4,20	55,33	5,45	8,97	58	53,53	7,25	11,93	63
Avg	67,33	63,63	3,70	**5,50**	61,34	5,99	**8,90**	47	59,18	8,15	**12,11**	48

Thermal Distribution," *International Symposium on Physical Design* , pp. 179-185, 1999.

[19] C.C. Chu and D.F. Wong,"A Matrix Synthesis Approach to Thermal Placement " IEEE TCAD, Vol.17, No11, pp. 1166-1174, 1998.

[20] A. Gupta, N, Dutt, F. Kurdahi, K. Khouri and M. Abadir,"Flooplan Driven Leakage Power Aware IP-Based SoC Design Space Exploration," CODES-ISS, pp. 118-123, 2006.

[21] Y. A. Cengel, *Introduction to Thermodynamics and Heat Transfer*, McGraw Hill 1^{st} Edition, 1996

[22] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh and s. Velusam "Compact Thermal Modeling for Temperature-Aware Design," *Design Automation Conference (DAC)*, pp. 878-883, 2004.

[23] R. H. Otten, "Automatic Floorplan design," *Design Automation Conference (DAC)*, pp. 261-267, 1982.

[24] D. F. Wong and C. L. Liu, "A New Algorithm for Floorplan," *Design Automation Conference (DAC)*, pp. 101-107, 1986.

[25] Y. Han, I. Koren and C. A. Moritz, "Temperature Aware Floorplanning," *Temperature-Aware Computer Systems (TACS)*, 2005

[26] D.A. Hodges, H. G. Jackson, R.A. Saleh, *Analysis and Design of Digital integrated Circuits,* McGraw Hill 3^{rd} Edition, 2004.

[27] D. Stroobandt, J. Depreitere and J. Van Campenhout, *Generating new benchmark designs using a multi-terminal net model,* "VLSI Journal", Vol. 27 pp. 113-129, July 1999.

[28] J.Cong and Y. Ding, *Flowmap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs,*"IEEE Trans. on Computer-Aided Design", Vol.13, pp 1-12, January 1994.

[29] E.S.Kuh and T. Ohtsuki, *Recent advances in VLSI layout,*" Proceedings of the IEEE", Vol. 78, pp. 237-263, February 1990.