Optimal Allocation and Placement of Thermal Sensors for Reconfigurable Systems and Its Practical Extension

ByungHyun Lee

School of Electrical Engineering and Computer Science
Seoul National University, Korea
frogfoot@ssl.snu.ac.kr

Taewhan Kim

School of Electrical Engineering and Computer Science
Seoul National University, Korea
tkim@ssl.snu.ac.kr

Abstract— **A dynamic monitoring of thermal behavior of hardware resources using thermal sensors is very important to maintain the operation of systems safe and reliable. This work proposes an effective solution to the problem of thermal sensor allocation and placement for reconfigurable systems at the post-manufacturing stage. Specifically, we define the sensor allocation and placement problem (SAPP), and propose a solution which formulates SAPP into the unate-covering problem (UCP) and solves it optimally. We then provide an extended solution to handle a practical design issue where the hardware resources for the sensor implementation on specific array locations have already been used up by the application logic. Experimental results using MCNC benchmarks show that our proposed technique uses 19.7% less number of sensors to monitor hotspots on the average than that used by the bisection based [1] approaches.**

I. Introduction

Aggressive scaling of process technologies has enabled very high logic densities on integrated circuits, which in turn lead to a sharp increase of power density, and thus the increase of die temperature. Recently, it was reported that the die temperature for commercial grade FPGAs typically reaches 80°C in a normal operation speed and reaches beyond 125°C in a high operation speed with excessively parallel execution [1]. Since high temperature on a chip causes serious effects on timing, power, reliability, and chip lifetime, it is highly desirable to maintain a low and even temperature distribution on the chip. However, due to the die size constraint, the physical cooling factors alone, such as heat sinks and fans, surrounding the die could not afford a complete solution. On the other side, controlling temperature through the control of operation executions (e.g., clock throttling [2]) is becoming an essential cooling factor. This run-time control of chip operation invariably requires thermal sensors to monitor the thermal behavior of the chip during the chip operation.

FPGAs are now one of the most suitable devices for creating high-density and high-speed reconfigurable systems, which can modify or adapt their functionalities to perform different tasks. Thus, monitoring and controlling the thermal status of the FPGA device during dynamic reconfiguration is very important. In this respect, this work focuses on the thermal sensor allocation and placement problem for reconfigurable systems on FPGAs. Note that for programmable logic arrays, the degree of the use of the programmable logic resources will be determined after when the target reconfigurable application is mapped. Thus, creating and placing thermal sensors at the manufacturing stage are inadequate. Instead, to minimize the waste of resource usage for thermal sensors as well as to effectively monitor the thermal behavior, the task of sensor allocation and placement should be considered at the post-manufacturing stage (precisely, after target logic mapping), at which the hotspot[1] locations of the reconfigurable logic were known.

It is well known that microelectronic delay increases as temperature increases. As a result, a way to estimate chip temperature is to construct an oscillator and calibrate its output drift in MHz per °C or °F.[2] Lopez-Buedo, *et al.* [4] used the oscillator-based (called *ring-oscillator*) sensors, combined with their auxiliary blocks (e.g., counting and control stages), for thermal monitoring in FPGA-based systems. The programmability in FPGAs gives a unique feature for embedding thermal sensors into target application. It allows the ring-oscillators to be dynamically inserted, moved, or eliminated by a system designer [5]. As another thermal sensor, the embedded thermal diodes on FPGA devices can be used. However, it is not possible to find, at the manufacturing stage, the best locations on the device at which thermal diodes are inserted since the same type of device can be used very differently according to the target applications. On the other hand, ring-oscillator has many advantages over thermal diode, except the high sensitivity to power supply variation, such as fully programmability and digital signals, and linear characteristics [4], [5].

In addition to the problem of finding best locations of sensor placement with minimum number of sensors, another issue to be solved is the 'practical' use of ring-oscillator in the reconfigurable systems. This issue arises due to the programmability of ring-oscillator. Even though the size of ring-oscillator is relatively very small (a sensor occupying around 0.3% of the Xilinx Virtex XCV800 model, including its auxiliary circuitry), placing a large number of sensors to monitor entire thermal behavior can cause a significant resource overhead, or may not be feasible due to the limited space for sensor implementation, especially when the reconfigurable logic is tightly fitted into the target device. For this reason, given a reconfigurable design exhibiting a certain thermal profile, it is quite important to minimize the number of sensors to be allocated and also find the best locations to monitor all hotspots.

In this work, we proposes a solution to the problem of thermal sensor allocation and placement for reconfigurable

[1] A *hotspot* refers to the location of chip at which the temperature exceeds a given temperature threshold T_{th}.

[2] This idea was originally proposed by [3].

978-1-4244-1922-7/08/$25.00 ©2008 IEEE

FPGA systems at the post-manufacture stage. Specifically, we define the sensor allocation and placement problem, and propose a solution which formulates the problem into the unate-covering problem and solves it efficiently and optimally. We then provide an extended solution to handle the practical issue where the hardware resource for the sensor implementation is limited. To the best of our knowledge, only the works in [6], [1] considered the problem of a 'systematic' sensor allocation and placement. Mondal, Mukherjee, and Memik [6] allocated and placed sensors one by one at a time in a greedy manner by finding a sensor position that covers the maximum number of hotspots. On the other hand, Mukherjee, Mondal, and Memik [1] enhanced the work in [6] by proposing a method called *recursive bisection* algorithm. Note that the bisection algorithm is another greedy method, and may lead to a sub-optimal results. Furthermore, the methodology used in [6], [1] shall not be applied in practice or insufficient in the sense that they assumed a sensor can be implemented in 'any' location on the fabric. In practice, some location may be occupied by the application logic. This implies that implementing sensor to a specific location at which a piece of application logic has already been mapped requires a (partial or entire) remapping of the application logic. However, this remapping will change the thermal profile of the target design, thus the hotspot distribution.

II. Thermal Sensor Allocation and Placement

A. Thermal Sensor

A direct technique for programming thermal sensors on FPGAs is based on ring oscillators. A ring-oscillator is a feedback loop that includes an odd number of inverters connected in a chain to generate the phase shifting at its output. Since temperature is one key parameter that is sensitive to the switching speed of transistors, the captured speed will be changed when the temperature of the circuit surrounding the oscillator is changed.[3] The two counters for the speed capture and the time allocation are included in the sensor. The speed capture counter is used to measure the ring-oscillator frequency and the time allocation counter is used to periodically enable the ring-oscillator when thermal sensing is needed.

We can fix a region on which a sensor can monitor temperature. The region will be a circle c_i (with radius a) or rectangle r (with size $l \times l$). A sensor s_i will be placed on the center of c_i or r_i. (Note that the value of a and l can be determined by using the formula in [7].) We call the circle c_i and rectangle r_i the *covered region* of sensor s_i. For simplifying our discussion, we use rectangles as covered region, as the works in [8], [1]. We call the value of l in the size specification of covered region r_i the *covering range*[4] of r_i. If the covering range l is large, the number of sensors to cover all hotspots will be reduced, but it loses the accuracy of approximating the covered region, and conversely if the covering range is small, it can accurately measure the thermal behavior on the covered region, but may need more sensors to cover all hotspots.

For implementing n sensors, n ring-oscillators are required, but the time allocation and speed capture can be shared. Nevertheless, even though mapping each ring-oscillator to FPGA requires at most 4 CLBs [5], as the value of n becomes large, the area, power, and thermal control overheads will be nontrivial. For example, Lopez-Buedo and Boemo [8] allocated 32 sensors to uniformly cover the fabric of XCV800HQ240-4C Virtex FPGA whose CLB array size is 56×84. Thus, the total number of CLBs occupied by sensors is at least 128, even not including the auxiliary logic overhead, such as time allocation, speed capture counters and mux and demux logic. Even if a grid-based uniform sensor placement is simple and is reasonably good to monitor the overall thermal behavior of the entire fabric, the number of sensors required increases as the CLB array size increases. Consequently, it is important to allocate and place sensors in a way to minimize the number of sensors allocated while every hotspot on the fabric is covered by at least one sensor.

B. An Optimal Sensor Allocation and Placement

Problem 1 *Sensor Allocation and Placement Problem* (SAPP): Given a set H of hotspots on FPGA and a covering range l, the problem is to find a set S of sensors and their locations on the FPGA such that it satisfies (i) for each $h_i \in H$, there is $s_j \in S$ that covers h_i by its covered region, and (ii) $|S|$ is minimum.

Our proposed sensor optimization algorithm, called SEN-opt, solves SAPP optimally by transforming it into the *unate covering problem* (UCP).

The UCP [10] is, given a matrix M of m rows and n columns, for which $M_{i,j}$ is either 0 or 1, is the problem of finding a minimum cardinality column subset C, such that for all C',

$$\exists_{j \in C'} M_{i,j} = 1, \forall i \in \{1, \cdots, n\} \Rightarrow |C| \leq |C'| \qquad (1)$$

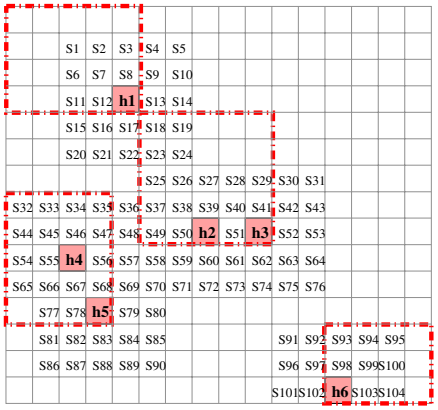
That is, the columns in the set C cover M in the sense that every row of M contains a 1-entry in at least one of the columns of C, and there is no smaller set C' which also covers M. The matrix M is called *constraint matrix* of UCP.

Thus, the transformation of SAPP into UCP is to construct a constraint matrix, as follows: Suppose we are given a $p \times q$ CLB logic array F of a configured FPGA with a set of hotspots (i.e., $H = \{h_1, h_2, \cdots, h_k\}$). SEN-opt then creates a matrix M of k rows and pq columns such that $M_{i,j}$ is set to 1 when the block that contains hotspot h_i is in the covered region of the (potential) sensor located at $F_{v,w}$ where $v = \lfloor \frac{j}{q} \rfloor$ and $w = j - v$, and set to 0, otherwise. For example, Fig. 1(a) shows six hotspots (h_1 through h_6) and possible sensor locations $(s_1, s_2, \cdots, s_{104})$[5] Fig. 1(b) shows the constraint matrix M corresponding to F in Fig. 1(a) where each of the blank entries is assigned with value 0. For example, we can see that hotspot h_2 can be physically monitored by any of the four sensor candidates s_{25}, s_{26}, s_{27}, and s_{58}.

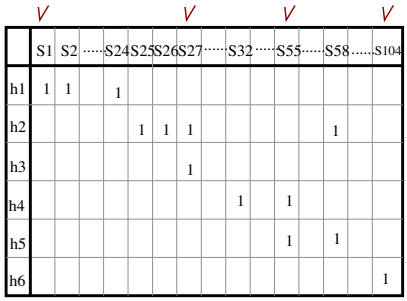
[3] For example, in [4] a frequency output of 45.5 MHz at 25°C was obtained using XCV50PQ240-4 FPGA in the experiments.

[4] In this work, we set l to 12, in terms of CLB blocks as [1] did.

[5] For brevity, some parts of sensor locations in F are specified in the figure.



(a) Hotspots and sensor candidates



V : cover solution

(b) constraing matrix of (a)

Fig. 1. An example showing the transformation of a problem of SAPP to an instance of UCP with the construction of constraint matrix of UCP.

Once the constraint matrix is constructed, SEN-opt applies UCP. The well-known procedure we adopt is the one in [11], which essentially solves the binate-covering problem (BCP) and is based on the branch-and-bound algorithm. The procedure employs a set of clever specialized strategies (e.g., row-dominance, column-dominance, computation of lower bound), which have been proven to be extremely useful, to systematically explore the search space. In the example in Fig. 1(b), SEN-opt produces $C = \{s_1, s_{27}, s_{55}, s_{104}\}$ and thus four (= $|C|$) sensors are the minimum to cover all hotspots. The covered regions of hotspots are shown in Fig. 1(a) as dotted boxes.

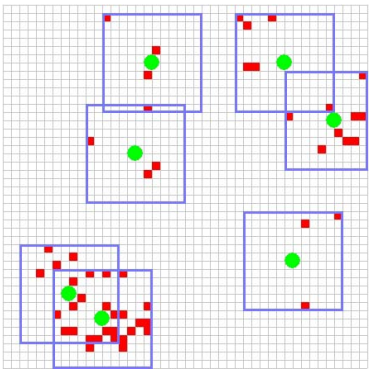
C. Practical Considerations

As mentioned in section II.A, the sensor creation requires a programmable resource. If there is a resource available in the location at which the sensor is to be inserted, there will be no problem. However, if there is no resource available or the resources are not enough, we need to move some of existing logic around the location to other locations. This means that remapping of the application logic is required. Since the space for sensors is very little in most of FPGA devices, the replacement will not cause a drastic change of placement of the application logic. However, even the small change of placement could affect a change of hotspot distribution, thus (partially) nullifying the feasibility of the prior result of sensor allocation and placement.

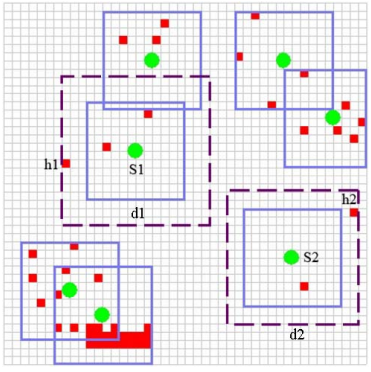
Let $S = \{s_1, \cdots, s_k\}$ and $L(s_i)$ be the sensors and the sensor locations produced by SEN-opt, respectively. Furthermore, suppose covering range l and hotspot set H, which are inputs to SEN-opt, are given. Then, our proposed design flow, called SEN-FLOW copes with the resource limitation for sensors by performing the following three steps:

• **Step 1** (*Replacement and identification of new hotspot set*): First, the sensors in S are programmed in their locations $L(s_i)$, and then the application logic is remapped. Then, the logic is run and a new hotspot set H' are identified. For example, Fig. 2(a) and (b) show the hotspot distribution before and after remapping of design alu4+apex4+ex5p+misex4 in Table II, respectively. As shown in Fig. 2(a), SEN-opt uses seven sensors to cover all the hotspots in H. However, when we assume 28 logic blocks (4 for one sensor) have been all used up, remapping of the application logic leads to slightly different hotspot distribution H' as shown in Fig. 2(b). (For mapping and remapping, we used VPR [13].)

• **Step 2** (*Extract uncovered hotspots*): We find a subset H'_{unc} of hotspots in H' that is not covered by any sensor in S with covering range of l. Then, for each hotspot h_i in H'_{unc}, we find the sensor in S whose corresponding covered region covers the hotspot. Let $d(h_i)$ be the minimum covering range of such covered region. Then we compute $l' = max\{d(h_1), d(h_2), \cdots\}$. For example, the computation of d_i and l' is specified in Fig. 2.



(a) Optimal sensor location by SEN-opt



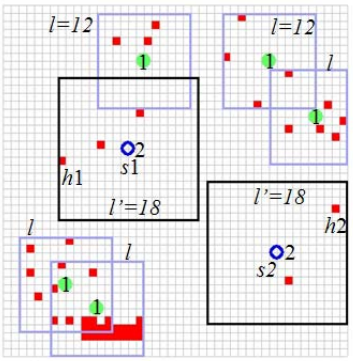
$l'=max\{d1,d2\}=d1$

(b) Logic remapping after sensor insertion

Fig. 2. A change of hotspot distributions before and after the sensor insertion for design alu4+apex4+ex5p+misex4 in Table II. (a) Finding the best sensor locations by SEN-opt, (b) remapping result by VPR [13] after the sensor insertion using the sensor locations in (a).

• **Step 3** (*Find minimum sensors covering the uncovered hotspots with the covering range l'*): We apply SEN-opt with

new covering range l', new hotspot set H', and the constraint of the candidate sensors being only those in S. Since $l' \geq l$ and each hotspot in H' will be covered at least one sensor in S, the solution obtained from SEN-opt will cover all the hotspots in H' by using a minimum number of sensors in S. For example, the nodes marked with blue circle on in Fig. 3(a) indicates the candidate sensors, and the table in Fig. 3(b) shows the corresponding constraint matrix. Then, by applying the unate-covering algorithm to the matrix we obtain the covering region, as indicated by the boxes with l' in Fig. 3(a).



(a) Covering h1, h2 using bigger covered regions

	S1	S2
h1	1	0
h2	0	1

(b) Constraint matrix of uncovered hotspots h1, h2

Fig. 3. Minimally updating the covered regions of sensors to cover uncovered hotspots h_1 and h_2 by logic remapping in Fig. 2(b).

Note that the three-step procedure takes into account two important factors: (1) *minimizing the number of sensors with* l' directly *reduces the error approximating the temperature*, and (2) *using only one more type of covering range* (i.e., l' only) rather than using multiple covering ranges is intended to *simplify the auxiliary logic*.

III. Experimental Results

We have implemented our proposed approaches SEN-opt and SEN-FLOW in C++ and a script, ran on a PC equipped with 2GHz AMD Athlon processor, and tested them on a set of benchmarks to assess how much they are effective. We have also implemented the existing approach Bisection with the "longest edge neighbor detect (lk-ng-det) heuristic in [1]. ([1] showed that lk-ng-det outperformed any other heuristics in terms of quality.) We evaluate our techniques in two-fold: (i) checking the effectiveness of SEN-opt on a set of benchmarks, in comparison with the results by existing techniques, and (ii) checking the effectiveness of SEN-FLOW on resolving the practical issue of the mapping conflict by the sensors and application logic.

• **Assessing the effectiveness of SEN-opt over existing techniques**: We tested SEN-opt and the best known existing methods Bisection [1] on the MCNC benchmarks in [12]. We used VPR [13] to map the netlists of the benchmarks into CLBs and route them. (Each CLB block used by VPR was set to contain four 4-input LUTs.) Then, the power model in [14] was used to calculate the amount of power consumption of the CLBs and nets based on the switching activities on the nodes. Then, the power numbers obtained were used to perform thermal simulation using the thermal simulator HotSpot [15]. We then extracted the hotspots from the simulation results by setting the temperature threshold T_{th}=85°C. The hotspot distribution for each benchmark was used as input to Bisection and SEN-opt. In addition, For all tested designs, we used a CLB array size of 50×50 because we found that the size was enough to map and route any of the designs. In the experiment, we set covering range l to 12 CLB-distance. Table I shows a comparison of results in terms of the number of sensors used by Bisection [1] and our optimal SEN-opt. *#hspot* represents the number of hotspots generated by HotSpot for each design. In short, Bisection is 19.7%away from our optimum.

TABLE I

Comparisons of the numbers of sensors allocated and placed by Bisection [1] and SEN-opt for the hotspots obtained using [13], [15] on CLB array size 50×50.

Benchmark	#hspot	# of sensors		red. over
		Bisec[1]	SEN-opt	Bisect
APEX2	20	4	3	25%
DIFFEQ	6	3	2	33%
CLMA	30	8	6	25%
S38417	46	12	9	25%
S38584.1	12	7	6	14%
ELLIPTIC	12	3	3	0%
EX1010	8	3	2	33%
FRISC	38	8	6	25%
PDC	17	4	4	0%
SPLA	29	6	5	17%
Avg.				**19.7%**

• **Assessing the effectiveness of our design flow** SEN-FLOW: We assumed that the application logic has been mapped to entire CLBs, thus there is no room for sensors even though the best locations to insert sensors are found. In fact, VPR [13]was very smart, so that maps logic very compactly, not generating any 'partially' used CLBs. Moreover, after a part of CLBs was remapped to sensor logic, the result of subsequent remapping of the target logic is very different even though the sensor logic maps to a small portion of the entire resources. One remedy we invented is to intentionally make the size of target logic almost close to the logic capacity of the CLB array. In that case, VPR responded with a little change in remapping. The designs listed in Table II are the ones that are close to the logic capacity of 50×50 CLB array.[6] where some netlists were obtained by combining multiple designs as indicated by '+' in the design names shown in the first column of the table. Since it is quite hard to compare our results with any other results fairly, we simply summarize the results produced by SEN-FLOW for reference. For example, for design compact, seven sensors are allocated. However, after remapping application logic due to sensor insertion, two hotspots among 43 are in the outside of the covered regions unless some sensors' covering length l is not set to a value greater than the initial value of 12. SEN-FLOW in fact

[6]Practically, we may exploit the feature: 'Commercial FPGA mapping tools have options and attributes to untouch some parts of mapping and routing result even when an additional remapping is performed.'

selected the minimum number of sensors from the existing sensors to cover the two uncovered hotspots h_1 and h_2. Here, two sensors among seven use $l' = 18$.

TABLE II
RESULTS OF SEN-FLOW FOR BENCHMARKS USING CLB ARRAY SIZE 50×50.

Benchmark	After SEN-opt #sensor	After remapping #hspot (cov.)	After remapping #hspot (uncov.)	After SEN-FLOW #sensor ($l = 12$)	After SEN-FLOW #sensor ($l > 12$)
CLMA	6	10	7	3	3 ($l : 20$)
S38584.1	6	11	7	3	3 ($l : 22$)
S38417	9	17	8	7	2 ($l : 20$)
EX1010+SPLA	7	9	1	6	1 ($l : 18$)
FRISC+SPLA	8	43	8	5	3 ($l : 28$)
FRISC+EX1010	8	41	15	4	4 ($l : 20$)
COMPACT	7	43	2	5	2 ($l : 18$)

compact = ALU4+APEX4+EX5P+MISEX4

IV. CONCLUSIONS

This work presented an effective solution to the sensor allocation and placement problem for reconfigurable systems when a ring-oscillator based sensor implementation at the post-fabrication stage was applied. The contributions of this work were (1) proposing a solution, called SEN-opt, to the sensor allocation and placement where SEN-opt *formulates the problem into the unate-covering problem and solves it optimally*, and (2) proposing a solution, called SEN-FLOW, to the practical issue caused by target logic remapping to make a programmable space for sensor insertion where SEN-FLOW fully utilizes the results by SEN-opt with the consideration of minimizing of the overhead of additional sensor monitoring circuitry while retaining the accuracy of temperature approximation as high as possible. From experimental results using benchmarks, we confirmed that our proposed approach could be used usefully to determine best sensor locations with minimal sensor allocation for practical design of reconfigurable systems.

ACKNOWLEDGEMENTS

This research work has been supported by Nano IP/SoC Promotion Group of Seoul R&BD Program, IT-SoC Program, ETRI project and System IC2010 project of Korea Ministry of Commerce, Industry and Energy. This work was also partially supported by the Ministry of Science and Technology/Korea Science and Engineering Foundation through the Advanced Information Technology Research Center.

REFERENCES

[1] R. Mukherjee, S. Mondal, and S. O. Memik, "Thermal sensor allocation and placement for reconfigurable systems," *IEEE/ACM International Confernece on Computer-Aided Design*, pp. 437-442, 2006.

[2] S. Gunther, *et al.*, "Managing the impact of increasing microprocessor power consumption," *Intel Technology Journal*, Vol. 5, February 2001.

[3] G. Quenot, N. Paris, and B. Zavidovique, "A temperature and voltage measurement cell for VLSI circuits," *European ASIC Conference*, pp. 334-338, 1991.

[4] S. Lopez-Buedo, J. Garrido, and E. Boemo, "Thermal testing on reconfigurable computers," *IEEE Design & Test of Computers*, Vol. 17, No. 1, pp. 84-91, 2000.

[5] S. Lopez-Buedo, J. Garrido, and E. Boemo, "Dynamically inserting, operating, and eliminating thermal sensors of FPGA-based systems," *IEEE Transactions on components and packaging technologies*, Vol. 25, No. 4, pp 561-566, 2000.

[6] S. Mondal, R. Mukherjee, and S. O. Memik, "Fine-grain thermal profiling and sensor insertion for FPGAs," *IEEE International Symposium on Circuits and Systems*, 2006.

[7] K. J. Lee, K. Skadron, and W. Huang, "Analytical model for sensor placement on microprocessors," *IEEE International Conference on Computer Design*, pp. 24-27, 2005.

[8] S. Lopez-Buedo, E. Boemo, "Making visible the thermal behavior of embedded microprocessors on FPGAs: a progress report," *Proc. International Symposium on Field Programmable Gate Arrays*, pp. 79-86, 2004.

[9] N. A. Sherwani, *Algorithms for VLSI physical design automation,* Kluwer Academic Publisher, Norwell, 1995.

[10] G. D. Hachtel and F. Somenzi, *Logic synthesis and verification algorithm,* Kluwer Academic Publisher, Norwell, 1996.

[11] V. Manquinho and J. P. Marques-Silva, "On using satisfiability-based pruning techniques in covering algorithms," *Design, Automation and Test in Europe Conference and Exhibition*, pp 356-363, 2000.

[12] S. Yang, *Logic synthesis and optimization benchmarks,* Microelectronics Center of North Carolina, 1991.

[13] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for deep-submicron FPGAs,* Kluwer Academic Publishers, 1999.

[14] K. K. Poon, *Power estimation for field programmable gate arrays,* Dept. of Electrical and Computer Engineering, University of British Columbia, 1999.

[15] K. Skadron, *et al.*, "Temperature-aware microarchitecture," *International Symposium on Computer Architecture*, pp. 2-13, 2003.