January 2008

On-chip Thermal Sensor Placement

Yun Xiang
University of Massachusetts Amherst

ON-CHIP THERMAL SENSOR PLACEMENT

A Thesis Presented

by

XIANG, YUN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2008

Department of Electrical and Computer Engineering

ON-CHIP THERMAL SENSOR PLACEMENT

A Thesis Presented

by

Xiang Yun

Approved as to style and content by:

_____
Wayne P. Burleson, Chair

_____
Sandip Kundu, Member

_____
Ramgopal Mettu, Member

_____
C.V. Hollot, Department Head
Department of Electrical and Computer Engineering

ABSTRACT

ON-CHIP THERMAL SENSOR PLACEMENT

SEPTEMBER 2008

XIANG YUN

B.S., ZHEJIANG UNIVERSITY

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Wayne Burleson

In the design of modern processors, thermal management has become one of the major challenges. Aggressive technology scaling and ever increasing demand for high performance VLSI circuits has resulted in higher current densities in the interconnect lines and increasingly higher power dissipation in the substrate. The importance of thermal effects on reliability and performance of integrated circuits increases as the technology advances. Thus a large number of thermal sensors are needed for accurate thermal mapping and thermal management.

However, a rise in the number of the sensors might lead to large area cost and increase the complexity of routing. So to accurately calibrate the thermal gradients and reduce the area cost for thermal sensors, a systematic sensor distribution and allocation algorithm is essential. In this paper, we will first look into the existing thermal sensor placement techniques and add some further optimization technique to an existing temperature related algorithm.

Then we will propose an algorithm based on the actual thermal gradient of the hotspots to determine the thermal sensor distribution on the microprocessors. The

algorithm is designed to find the minimum number of sensors required and their correspondent locations while ensuring that all the calibrated temperatures' deviation will be within a pre-determined error margin. We have used QT clustering algorithm to partition the hotspots and implemented a novel scheme for fast and efficient sensor location calculation. Our simulation is set on processor alpha21364, also known as EV6. Moreover, the anisotropic property of heat dissipation on chip is also considered.

The final piece of this work will lead to some thermal management techniques on chip given the knowledge of the thermal sensor distribution. An optimized monitor network on chip (MNoC) scheme is set up for the non-uniform distributed sensors returned by the proposed thermal sensor placement algorithm. Two different schemes are proposed and analyzed to address this problem.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

As the technology continues scaling down, the power consumption of the modern processors increases significantly as chip size getting larger, feature size becoming smaller and frequency running higher. It is reported that the thermal gradient can reach as high as 50°C across high performance microprocessor substrates[17][18] and the situation could be even worse when we enter the GHZ frequency regime. Large temperature variation across the chip will decrease the reliability of the circuits as well as degrade its performance. It can not only cause timing errors, but also inflict physical damage to the circuit because of the phenomenon of electron migration. As a result the lifetime of the circuits will be greatly reduced. The thermal effect will also increase the interconnect delay and worsen the interconnect leakage, which will restrict the performance of the circuits. It is reported in [19][20] that the Elmore delay will increase by 5% if the temperature is raised by 10 degrees.

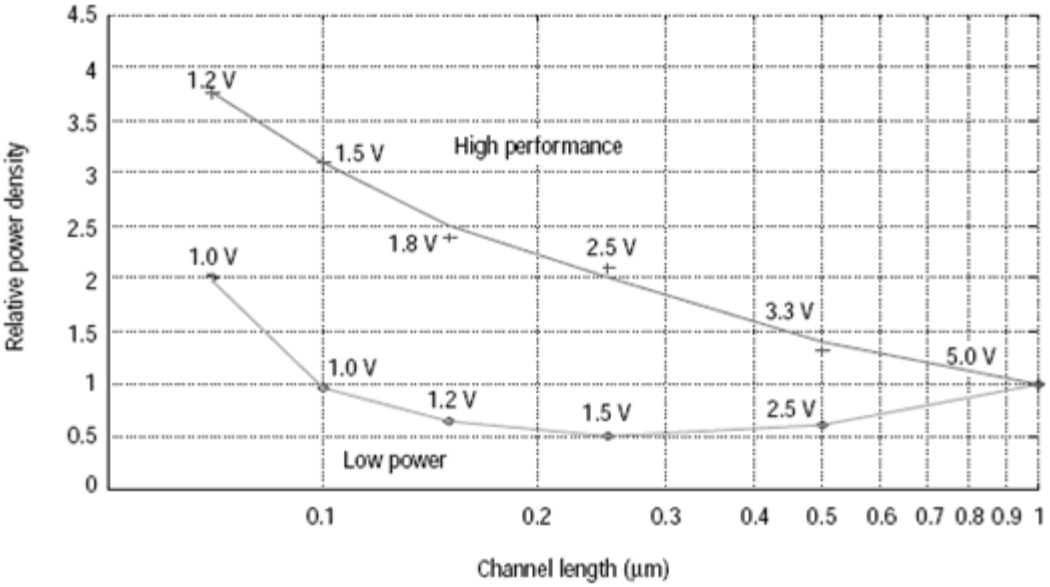
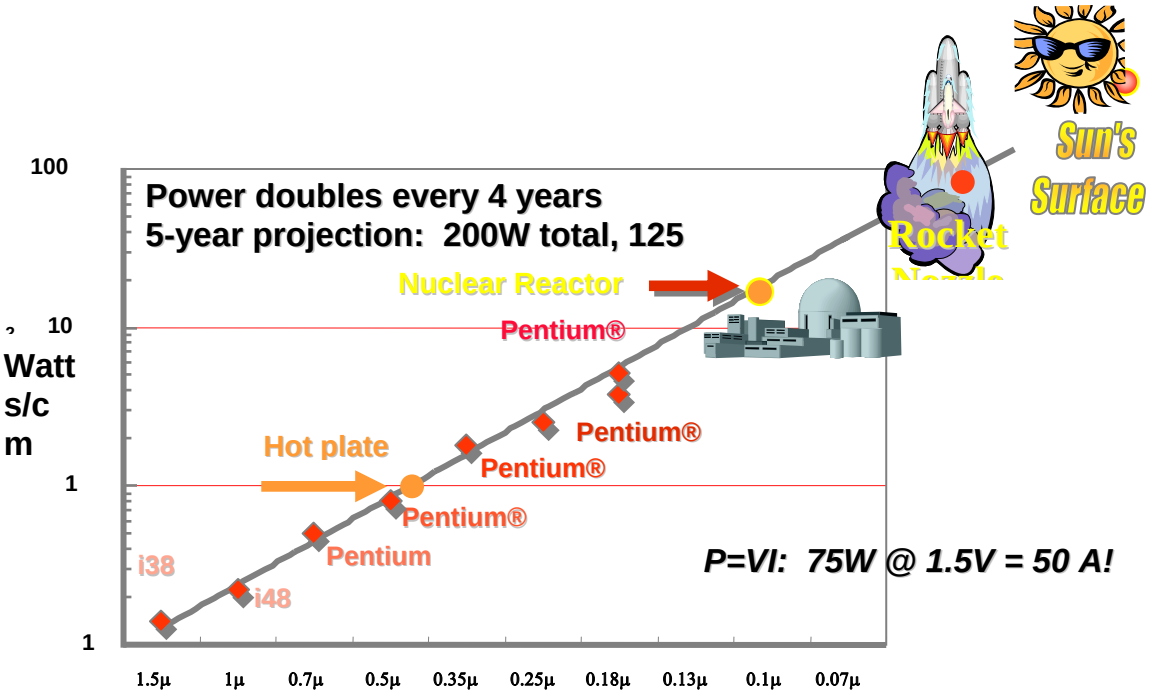


Figure 1. Expected power density as a function of MOSFET length

But as technology downscaling leads to greater power and package densities as shown in figure 1, thermal effects become significant in more and more designs. As it is stated in[26], Industry analysts suggest that thermal effects will be very important at dimensions under 0.1 microns (figure 2). At this level, increased power densities result in higher chip temperatures, changes in device parameters, and eventually less reliable circuits.

The hottest region of an electronic assembly is always the chip surface. An unfortunate placement of components or die-attaching errors can cause undesirable peaks, or hot spots, in temperature distribution. A hot spot can reduce the reliability, shorten the lifetime, or even immediately burn out the chip.



* "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies" – Fred Pollack, Intel Corp. Micro32 conference key note - 1999. Courtesy Avi Mendelson, Intel.

Figure 2. Power consumption road map by Intel.

Realizing the crucial impact of the thermal effect to the circuit, modern high performance processors employ advanced techniques for thermal management, which rely on accurate readings of on-die thermal sensors. With current technology scaling and integration trends, timely and accurate detection of localized heating will be of evermore importance. As the significance of thermal effects on reliability and performance of integrated circuits increases, careful planning and embedding of thermal monitoring systems will be essential. Thermal monitoring using on-die thermal sensors provides the means to assess the run-time thermal profile of the system. Also, in cases where the number of sensors is limited and we have multiple objectives to observe which beyond the capability of the current on-die sensors. We can use the conception of "virtual

sensor"[32], which is a runtime temperature sensing methodology that can be used for high performance architecture studies on real hardware, to estimate the thermal behavior of the chip. The control unit receiving input from the hardware sensors can effectively minimize the number of times the temperature values are calculated, and hence the computation overhead of the software can be reduced. While in addition, the system can obtain a detailed spatial distribution of temperature through software modeling even in the absence of fine-grained on-chip sensors.

Typically, in the Dynamic Thermal Management mechanisms employed by high performance microprocessors, thermal sensors trigger alerts if the junction temperature exceeds a specified limit. Based on these alerts, the processor power consumption is regulated via clock throttling. Localized heating on a processor is application dependent and different architectural choices result in diverse thermal profiles. We can use a hybrid hardware-software solution to obtain an accurate and detailed temperature sensing method that can be used for DTM with less cost. Compared to a purely hardware-based or software-based solution, this sensor-fusion algorithm can reduce the overall cost and provide the benefits of both approaches. Moreover, the specific purpose of thermal monitoring determines which specific form of sensing setup should be used. If the target is design of emergency intervention mechanisms, then the spots that reach the highest temperatures on the processor are most relevant. If we are interested in controlling temperature dependent leakage power and developing dynamic mechanisms, we need to capture the thermal profile of individual components even if the absolute temperatures are not threatening reliability. Similarly, if temperature dependent delay violation is the

focus, then the thermal condition of the critical path of the processor must be monitored closely.

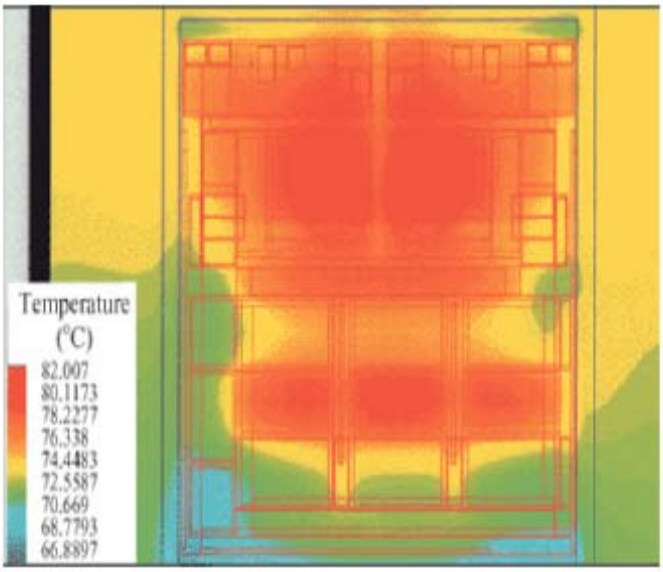


Figure 3. Map of FET junction temperatures for 115W packaged Power4 chip [IBM]

Ideally, a chip's temperature must be monitored at fine granularity using as many sensors as needed. However, most sensors are based on analog CMOS circuit designs. Precise temperature measurement typically requires matched transistors, and as process variations grow in severity, accurate sensors may require large devices to compensate. This in turn increases their size and power requirements. On-chip sensors can also be difficult to calibrate due to both physical and environmental variations, and adding sensors may therefore increase testing costs. Most current CPU designs use minimal number of sensors (with the exception of IBM POWER5 that makes use of 24 on-chip sensors). With limited number of available sensors, their placement becomes important for reliability of the thermal map generated by the sensors. Malicious programs could cause overheating in units outside the reach of the placed sensors and cause permanent

damage. The choice of thermal safety margins also dictates sensor number and placement; smaller margins provide high fidelity in the reported thermal map but require more sensors.

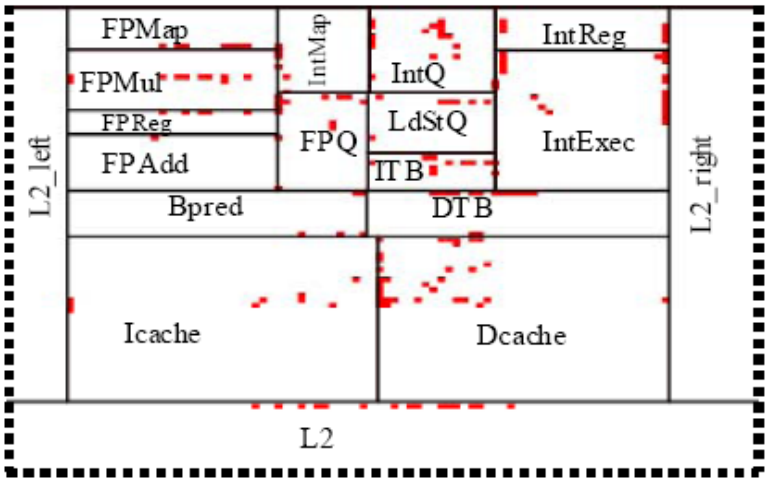


Figure 4. Distribution of Hotspots for each processor block for SPEC2000 benchmarks[3]

Different architectures with different workloads have reported different functional blocks to be the 'hotspot' prone blocks. In the Intel Pentium 4 processor, the single thermal sensor is placed near the rapid integer unit [1]. Skadron et al [2] report that in the Alpha 21364 architecture, the Register File appears to be the hottest component. Thermal analysis done by Memik et al [3] identified the Instruction Queue as the block generating the hottest points consistently. The location of the hottest point within each block shifts across applications. Thus, the number of distinct hotspots and their relative spatial distribution within each individual block can present unique characteristics.

Several local optimization techniques applied to various processor components such as "Heat and Run" thread assignment for multiprocessors [4], activity migration to reduce hotspots [5], thermal aware task allocation and scheduling[21], temperature-aware steering, clustering and thermal-aware renaming and committing mechanisms have

been proposed. These require accurate local thermal monitoring at much finer granularity (component level). Since the existing of the hotspots follows the principal of locality(both spatial and temporal), a strategic placement of thermal sensors becomes essential to support fine-grain dynamic optimizations as well as reduce the cost of area and complexity of interconnect routing.

1.2 Related Sensor Placement Techniques

1.2.1 Art Gallery Problem

The thermal placement problem is similar to the art gallery problem in some certain perspectives. The art gallery problem for a polygon P is to find a minimum set of points in P such that every point of P is visible from some point of G(figure 5). This problem is shown to be NP-hard by Lee and Lin.

As stated in[24], a polygon which shown in figure5 is generally defined as an ordered sequence of at least three points: v_1, v_2, …, v_n in the plane, called vertices, and the n line segments $\overline{v_1v_2}$, $\overline{v_2v_3}$,…, $\overline{v_{n-1}v_n}$ and $\overline{v_nv_1}$, called edges. A simple polygon is then a polygon with the constraint that nonconsecutive edges do not intersect. A simple polygon is a Jordan curve, and thus divides the plane into three subsets: the polygon itself, the (bounded) interior, and the (unbounded) exterior. However, we will henceforth use the term “polygon” to refer to “simple polygon plus interior.” Polygons are thus closed, bounded sets in the plane.

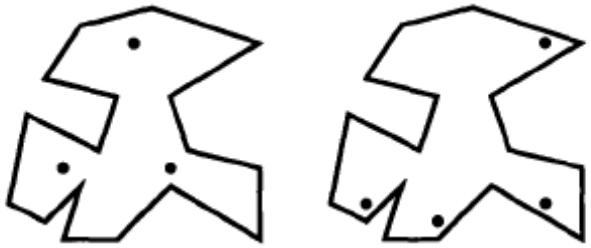


Figure 5. Polygon and guard in the art gallery problem.[24]

A guard set G is said to cover a polygon *P* if the collection of sets $\{V(g,P) \mid g \in G\}$ covers *P*. The points in the polygon on the left in Fig. 3 are a covering guard set. We will later see that the definition of covering for guard sets is a simple generalization of the usual definition of covering given above. The ***art gallery problem*** for a polygon *P* is to find a minimum-cardinality covering guard set G for *P*. It is so called because one envisions the polygon *P* as the floor plan of an art gallery, and the points of G as locations to place guards, ***so*** that every part of the art gallery is seen by at least one guard. We use *g(P)* to denote the cardinality of a minimum covering guard set for the polygon *P*. The original art gallery problem, posed in conversation by Klee to Chvital, is to find the smallest number of point guards necessary to cover any polygon of *n* vertices; this number will be denoted *g(n)* (not to be confused with *g(P)* as defined above). In terms of galleries, *g(n)* is the minimum number of guards necessary to supervise any gallery with *n* walls.

There exists a close resemble between the sensor placement problem and the art gallery problem (AGP) addressed by the art gallery theorem. The AGP problem can be informally stated as the determination of the minimum number of guards required to cover the interior area of an art gallery. However, our problem here is a bit different from the art gallery problem in definition. In our application, the objective is fixed and has a

certain sensing range, which means that a guard needs to be close enough to discover them. And if the distance between the objective and the guard is too far away, then the chance of it going undiscovered will be very high. Similarly, this revised version of AGP should also be NP-hard.

1.2.2 Sensor Placement Problem in Other Areas

In other sensor placement areas such as distributed wireless sensor network(DWSN), a lot of research has already been done to deploy the sensor network less costly and more efficiently. For example, a power constraint sensor placement algorithm is proposed[29] to explore the problem of deploying a finite number of sensor nodes in a geographic area under power consumption constraints.

Also in [30], a simulated annealing based algorithm is introduced to address the optimization problem. Simulated annealing (SA) is a highly reliable method for solving hard combinatorial optimization problems. Initially, it is assumed that the sensors are deployed at all grid points. In each loop, an attempt is made to remove one sensor if the cost constraint is not met. Otherwise, an attempt is made to move a sensor to another randomly chosen position. Moreover, the stopping criterion is modified to improve efficiency. Besides frozen temperature, t_f, is reached, when both complete coverage and discrimination are achieved, the procedure will then be stopped. The solution with complete coverage and discrimination may not be optimal. However, the solution is the desired solution to this problem and the proposed algorithm is very effective, scalable, and robust. We will use the conception of this optimization technique in our QT clustering algorithm.

CHAPTER 2

THERMAL SENSOR PLACEMENT TECHNIQUES

2.1 Thermal Gradient Model

The sensor placement problem has been explored with different motivations and perspectives. As a result, various contributions have been reported over the years. Lee et al [6] presented an analytical model that describes the maximum temperature variation between a hotspot and a region of interest. In this work, it is proposed that the information about the maximum possible temperature differential between a hotspot and a potential sensor location can be used to determine the safety margins for sensor placement and also, may help in determining the optimal number of sensors needed to detect the hotspots. If the temperature at the local hotspot is assumed to be the highest, then at a distance 'r' from it, the temperature difference between the 2 points is given as $T_{src}(r)$.

$$T_{src}(r) = T_{src-\max} \times (1 - e^{-\frac{2r}{K}}) \quad (1)$$

The constant 'k' denotes the thickness of the processor's package-die, heat spreader and thermal interface material. The thickness of each packaging material is multiplied by a factor based on the thermal resistivity of the material and that of silicon. When there is a single power source, the temperature drops off exponentially as the distance from the source increases. When the source is at its local maximum power density, the value of $T_{\max}$ can be derived as the difference between the maximum and minimum temperature value. However, in reality multiple power sources can be expected

to be active at the same time. The effects of different power sources are assumed to be independent of each other and follow a linear relation allowing superposition. Thus, for multiple power sources, $T_{src-\max}$ is derived as follows:

$$T_{src-\max} = \sum_{i \in Units} (T_{i-\max} - fa \times T_i(r_{src} - i)) \quad (2)$$

Generally speaking, if we have already had the temperature information of the hotspots and cold spots on chip, $T_{\max}$ can be derived by simply substrate the temperature of hotspots by that of the cold spots.

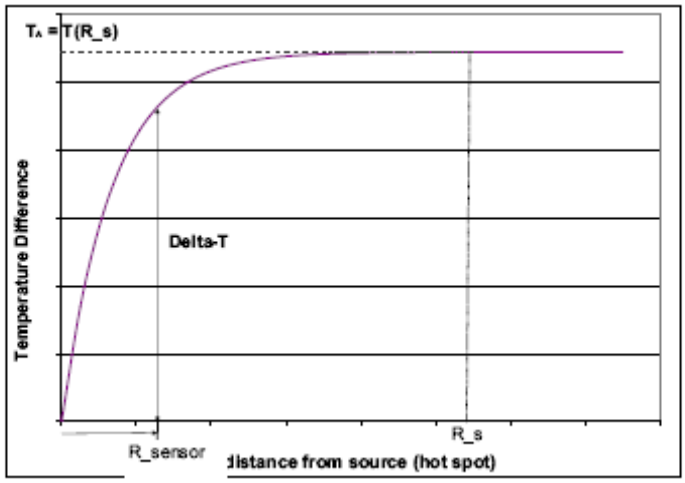


Figure 6. Maximum precision error versus distance.[6]

From figure 6 we can see that the precision error between the hotspot and the sensor location is a function of distance. And the temperature error will increase as the distance increases. Based on these observations, it would be possible to find an optimal distribution of thermal sensors for accurate calibrations since the hotspots tend to cluster according to the principle of locality as described in [22].

2.2 Geometric Sensor Placement Techniques

2.2.1 A Module Based Sensor Placement Algorithm

Gunther et al [7] present observations on thermal maps and points to opportunities for optimized decisions on sensor placement. Bratek et al [8] present sensor placement for fault diagnosis of integrated circuits, by linking temperature sensors and power modules in pairs as shown in figure 7 .

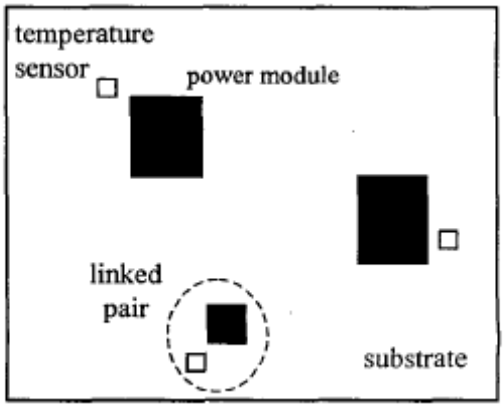


Figure 7. Temperature sensors and power modules in pair.[8]

In their approach, the principal for sensor placement is to locate the sensor near the power module that connected with them while keep it as far as possible from the other power modules. The location of a sensor depends on all the power modules on chip. However, Skadron et al[6] has discovered that the hotspots of the chip will shift as different applications are applied by the microprocessor. So just simply assigning one sensor to a block will not produce a thermal map that is accurate enough.

2.2.2 Dynamic Placement Technique

In the FPGA area, current technologies limit maximum temperature to 125 centigrade degrees. Current solutions, like the embedded diode available in Xilinx Virtex

and Altera Statrix FPGAs, or the use of infrared cameras only solve partially the problem. In the first option, the temperature is measured in only one point of the die, so the information derived can only be useful to know that there is a problem, but not to identify it. On the other hand, infrared cameras require the direct vision of the silicon, and thus, they can not be employed in actual working conditions.

Boemo et al [11][25] have developed a new thermal monitoring strategy suitable for FPGA-based systems. Their main idea is that a thermal sensor can be dynamically inserted, operated and eliminated from the circuit under test using run-time configuration. A ring-oscillator together with its auxiliary blocks is first placed in the design. After the actual temperature of the die is captured, the value is read back via the FPGA configuration port. Then the sensor system is eliminated from the chip in order to release programmable resources and prevent self-heating. The novelty of the approach lies in dynamic reconfiguration of the sensor system with lesser focus on the actual physical placement of the sensors.

Here we have the idea of reconfigurable sensors. The sensor can be dynamically inserted and deleted from the circuit. However, there are some deficiency and question here:

1, it needs a carefully scheduled plan. We need to precisely predict the location of hotspots and then correspondently insert a sensor nearby.

2, it would take time to reconfigure the circuit. And the sensors' accuracy is limited.

3, reconfigurable ability costs overhead. In Boemo's experiment[25], the method requires a reconfiguration of the blocks to the thermal sensors each time the temperature is needed

to be calibrated, hence it will reduce the performance of the circuit, as well as increase the area occupation of the chip.

2.2.3 Hotspots Clustering Based Placement Algorithm

Memik's group at Northwestern University has proposed several sensor placement algorithms for both microprocessors and FPGA-based systems. Their approaches are mainly based on the known hotspots of some typical benchmarks. The assumption here is that the currently known hotspots indicate the possible locations of the hotspots for future applications according to the principal of locality. The assumption is reasonable and could be verified by the correspondent experiments.

In [9] and [23] Memik et al proposed a recursive bi-partitioning based sensor placement algorithm that addresses the creation of a resource efficient sensor architecture for computing systems that are of regular nature, such as logic array-based computing platforms. The algorithm minimizes the number of thermal sensors needed to embed in a regular structure and determines sensor locations required to maintain a given accuracy in temperature sensing for a given design. Targeting FPGA systems as its platform of implementation, the algorithm divides the FPGA area into an array of configurable logic blocks and then assigns sensors in such a way that every hotspot is covered by some sensor and the number of such sensors needed is minimized.

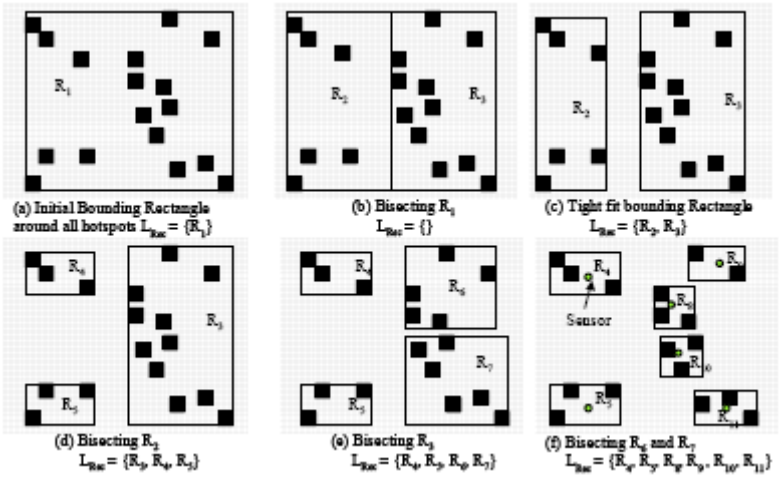


Figure 8. Stages of the Bisection Function of the hotspots.[9]

As shown in figure 8, each sensor has a coverage area dimension C_D. A sensor can monitor hotspots that lie within its coverage area of height and width at-most equal to C_D. The algorithm attempts to create a set of rectangles that cover the chip area, each having a height and width at most C_D, containing a set of hotspots. The sensor is placed at the center of a bounding rectangle which guarantees that all the hotspots within the rectangle are within the range of the sensor. The algorithm works by recursively creating bounding rectangles around hotspots and bisecting them until both height and width are less than or equal to C_D. A key element of the algorithm is that it prefers to have majority of the hotspots concentrated in one of the bisected rectangles rather than sparsely distributing hotspots in both bisections. This leads to a higher coverage (of hotspots) by a fewer number of sensors. The detail flow of the algorithm is shown in figure 9.

Minimal Sensor Placement Algorithm

Input: Dimension (Rows, Cols) of the CLB Array
Set of hotspots H and positions of hotspots $h_i(x_i, y_i)$, $h_i \in H$
Coverage area dimension C_D

Parameters: EdgeSelectType, BisectType

1. Create initial bounding rectangle R_{Init} for all the hotspots
2. Create a list of bounding rectangles L_{Rec}
3. Push back R_{Init} onto L_{Rec}
4. List iterator $L_{iterator} = L_{Rec}.begin()$
5. **While** ($L_{iterator}$!= $L_{Rec}.end()$)
   a. Get R_i pointed by $L_{iterator}$
   b. If (Checksides(R_i) = true) { Increment $L_{iterator}$; Continue;}
   c. Bisect(R_i, BisectType, EdgeSelectType)
   d. Assign hotspots of R_i to R_{i1} and R_{i2} to L_H^{i1} and L_H^{i2} respectively
   e. Create tight rectangles R_{i1}' (R_{i2}') containing L_H^{i1} (L_H^{i2})
   f. Push back R_{i1}' and R_{i2}' into L_{Rec}
   g. Delete R_i from L_{Rec} and Increment $L_{iterator}$
   End while
6. Size of the set of sensors S n = Size of L_{Rec}
7. Position of s_i = center of r_i, where $s_i \in S$, $r_i \in L_{Rec}$, $\forall i \in \{1 \ldots n\}$

Output: Number and Position of a set of sensors which covers H

Figure 9. Sensor placement algorithm flowchart.[10]

In [10] Memik et al proposed a variant of the above approach where the range of the hotspots is determined by using Skadron-Lee's analytical model described in [6]. The algorithm creates a set of circles where each circle is centered on a hotspot and has a radius equal to its range as shown is figure8. The radii of the circles are reduced to factor in a safety margin to approximate each circle with the CLB that lie fully within the circle. The CLB which covers the maximum number of hotspots is chosen for sensor placement. Identifying such a CLB ensures that a sensor placed in such a CLB can accurately sense temperature of a maximal number of hotspots (for which distance between the hotspot and sensor is less than its sensitivity range so that the calibrated temperature will be within the predetermined error margin). An assumption made in this approach is that a single CLB will be capable of implementing a thermal sensor. If one such is not found then 2 remedies are proposed:

(1) to find an unused CLB in close proximity

(2) local remapping of the design to create an unused CLB for sensor placement

Or in worst case we can add some extra sensors in the unused CLBS at the cost of area.

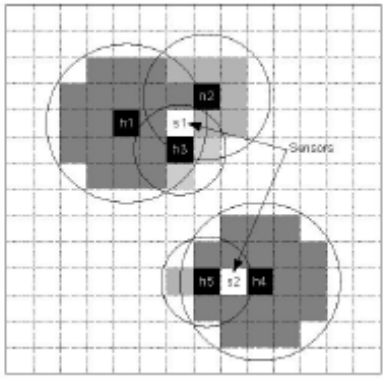


Figure 10. Example of dynamic thermal sensor insertion[9]

CHAPTER 3

A TEMPERATURE BASEDTECHNIQUE AND ITS IMPROVEMENTS

3.1 A Temperature Based Sensor Placement Algorithm

Memik et al. in [3] has proposed a k-mean clustering algorithm based on hotspots clustering for microprocessors to allocate hotspots into different groups and assign a sensor to monitor each group. The algorithm identifies an optimal physical location (referred to as the 'cluster center') for each sensor such that the sensor's attraction towards steep thermal gradients is maximized based on different programs running on the processor.

K-mean clustering can be defined as: given a set of *n* data points in *d*-dimensional space, determining *k* centers, such that the mean square distance from each data point to its nearest center is minimized. The k-mean clustering works by iteratively refining the position of the k cluster centers. Initial cluster centers are assigned to any *k* points from *n* elements and the membership set (based on the Euclidian distance) of each cluster is calculated. For number of sensors equal to *k*, *k* clusters need to be created to monitor *n* hotspots. The *k* cluster centers corresponding to each of the *k* clusters specify the location of a thermal sensor. The goal of the algorithm is to have all the hotspots assigned to clusters such that the Euclidian distance between the hotspots in a cluster and the cluster center/sensor location is minimized.

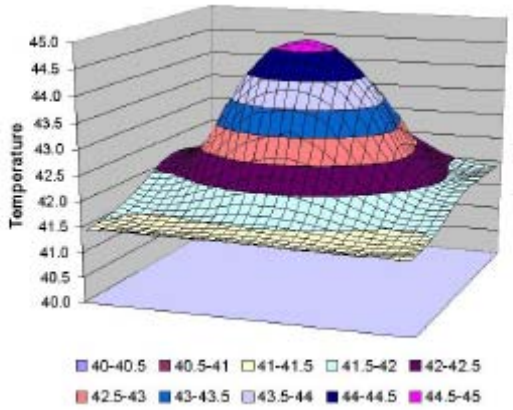


Figure 11. Temperature profile in 3D space[10]

In their approach, instead of a 2D implementation they use a 3D structure to determine the clusters as shown in figure9.

$$E(O_j, h_i) = (O_{ix} - h_{ix})^2 + (O_{iy} - h_{iy})^2 + (O_{it} - h_{it})^2 \qquad (3)$$

Equation(3) is used to calculate the distance between then hotspots and sensor in the 3D space. Using 3-dimensional distance evaluation is more likely to create clusters containing hotspots having similar thermal characteristics like temperatures. Therefore, a 3rd coordinate that indicates the temperature of the hotspot is also used for cluster formation. Determination of physical sensor location within the clusters is guided by moving the cluster center to the relatively higher temperature hotspots.

$$O_{x,y} = O_{x,y} + h_{x,y} - \alpha(O_{x,y} - h_{x,y}) \times (h_t - O_t / size(M)) \qquad (4)$$

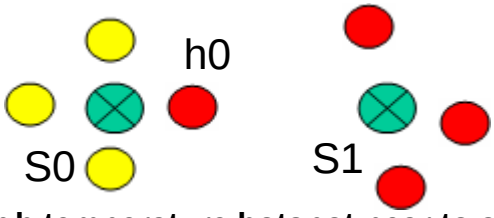
$$O_t = O_t + h_t \qquad (5)$$

Here $O_{x,y}$ and $h_{x,y}$ represent the coordinated of the sensor and hotspots respectively, O_t and h_t represent the temperature of the sensor and correspondent hotspots. Specifically equation(4) and equation(5) are used to determine the locations of sensors for a cluster.

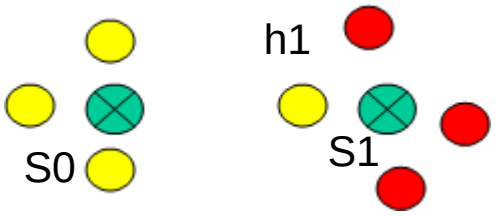
To optimize the location of the sensor, an attraction coefficient α is used that causes the cluster center to move away from a hotspot if its temperature is lower. In the original approach, a universal value is assigned to alpha while in our approach we will use different α for different blocks to maximize the optimization result.

3.2 Pitfall of the current 3D approach

The 3D k-mean clustering algorithm works well under some conditions. However, because of the complexity of the practical hotspots distribution, it might not always be able to find the optimal solution in every case. Moreover, in some certain circumstances, the 3D approach could even undermine the overall performance of the algorithm. Figure10 and figure11 shows examples of such situations.



1, High temperature hotspot near to a low temperature cluster



2, Low temperature hotspot near to a high temperature cluster

Figure 12. Performance of the 3D algorithm under different circumstances

In the figures above, the green spots represent the sensors, the red spots represent the hotspots with a relatively high temperature and the yellow dots represent the hotspots which have a relatively low temperature. Thus in figure12.1, we assume that h1 is the hotspots beside sensor S1 with a low temperature. And we can see that the other hotspots around S1 have a relatively high temperature. According to the 3D k-mean clustering algorithm, here we assume that h1 is included in cluster S0 rather than S1 because of its thermal characteristics. As Skadron et al claimed in[6], the temperature droop around a hotspot can be roughly estimated by $T_{src}(r) = T_{src-\max} \times (1 - e^{-\frac{2r}{K}})$, which means that if a hotspot has a higher temperature, the thermal gradient around it will be steeper. So in this case, S1 has ruled out the influence of the low temperature hotspot h1, its location will be more close to the high temperature hotspots, thus as a result the overall performance will be improved.

However, when we look at figure12.2, it tells another story. h0 represents a hotspot with a high temperature found near to sensor S0 which is surrounded by mostly low temperature hotspots while far away from sensor S1 which is near to a bunch of high temperature hotspots. In this case, again according to its thermal characteristics, h0 is attributed to cluster S1. So as a result, in order to balance the location between the hotspots inside the cluster, S1 will be driven away from the other hotspots with high temperatures. The overall performance will drop significantly under such circumstances since the performance gain of sensor S0 will be much too small compared to the performance loss of sensor S1.

The analysis tells us that the transformation from temperature to distance in the 3rd dimension should be handled with great care. It is very important to the overall

performance of the algorithm. Thus besides the work already done, we also need to optimize O_t and h_t in equation(4) and equation(5) which will be introduced in the next section.

3.3 Improved Version of the Algorithm

From the analysis in part 3.2, we can see that the 3D k-mean clustering algorithm will not always bring the most optimal distribution of sensors. Especially, in the 3D algorithms we use the temperature as the Z coordinates, which equals that we transfer the temperature to distance and we must be very careful about such transformations. The criterion that we transfer temperature to distance affects the formation of the clusters. If 1 degree of temperature transfers to too little distance, we shall lose the advantage of the 3D algorithm. However, if 1 degree temperature stands for a large distance in Z coordinate, in the end the high temperature hotspots which are too far away to each other will be included in the same cluster, and as a result the overall performance will be undermined significantly.

In the original design, the absolute value of the temperature is used for the Z coordinate transformation. It is not very reasonable in many cases since in a typical chip the distance between 2 hotspots is typically a few hundred microns at most while a precision error of larger than ten degrees is quite common. By implementing such policy of transformation, the temperature becomes the dominant factor to determine the clusters. We just simply group the hotspots with similar temperature features together and find the sensor location accordingly. This will not lead to an optimal solution in many cases. Thus in our version of 3D k-mean clustering algorithm, we will add a new coefficient β here

to optimize the algorithm. Now we use equation(6) to determine the distance in the 3d space.

$$E(O,h) = (O_x - h_x)^2 + (O_y - h_y)^2 + [\beta(O_t - h_t)]^2 \qquad (6)$$

And in this thesis, we shall use about seven different stages of values for β (from 0.00000001 to 0.1, multiplied by 10 between 2 consequent stages). It is true and obvious that if we increase the value of β we tried, it is more probably to increase the performance of the algorithm. However, we need to balance performance and the time cost of the algorithm. If we give n stages to the Beta value, the running time will be multiplied by n, thus constrains our choice of number of stages here. This improved version of 3D k-mean clustering algorithm will be used to compare with our own algorithm.

CHAPTER 4

QT CLUSTERING BASED THERMAL SENSOR ALLOCATION ALGORITHM

4.1 Problem Formulation

Skadon et al[6] has proposed equation(7) as stated below to find the radius R around a hotspot in order to keep the precision error with a certain degree $\triangle T$.

$$R = 0.5 \bullet K \bullet \ln\left(\frac{T_{\max}}{T_{\max} - \Delta T}\right) \qquad (7)$$

The equation implies a circular region around the hotspots for thermal sensor placement to keep the precision error within $\triangle T$. We will use this property to explore other possible solutions for thermal sensor placement optimization problem.

Moreover, according to equation(7), since $T_{\max}$ stands for the difference between the maximum and minimum temperature value, it is implied that the thermal gradient around the hotspots is not decided by the temperature of the hotspot alone. The overall thermal distribution across the chip must also be taken into consideration.

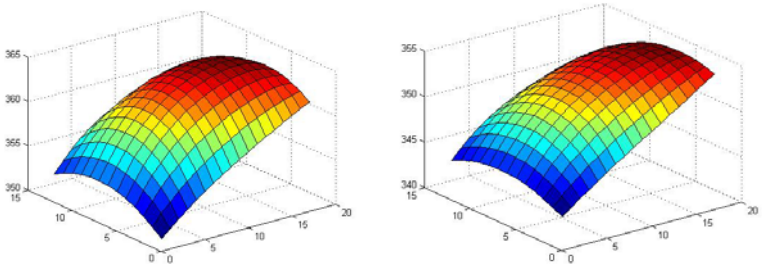


Figure 13. Thermal distribution of area around 2 hotspots

Figure13 shows the thermal distribution around 2 different hotspots. We can see that although their temperatures are differed by more than 10 degrees, the thermal gradient around them are similar since they are very close in terms of $T_{\max}$. Thus the method Memik's group used has some deficiency since they just use absolute temperature of the hotspots for optimization.

Our goal is to systematically analyze the thermal maps to identify sensor locations, which lie close to a maximal number of eventful thermal spots across a range of applications. We formulate this problem as a clustering of the points of interest in the spatial domain. A sensor is allocated to each cluster and this sensor will monitor the points associated with that cluster. We will find an algorithm to partition the hotspots and allocate them to different clusters. We also need to find a way to determine the location of sensor for each cluster for the purpose of further optimization. Our proposed algorithm will be based on QT clustering and we hope to find a sensor distribution with a minimum number of sensors required while in the mean time ensuring that the precision error of all the hotspots is within a certain error margin $\triangle T$.

4.2 A QT Clustering Based Algorithm

4.2.1 QT Clustering Algorithm

The QT clustering algorithm can be defined as[12]:

QT (quality threshold) clustering (HEYER et al, 1999) is an alternative method of partitioning data, invented for gene clustering. The algorithm is:

1, The user chooses a maximum diameter for clusters.

2, Build a candidate cluster for each point by including the closest point, the next closest, and so on, until the diameter of the cluster surpasses the threshold.

3, Save the candidate cluster with the most points as the first true cluster, and remove all points in the cluster from further consideration.

4, Reiterate with the reduced set of points

In our application, the hotspots in the thermal map are defined as 'points' to be classified into clusters. And According to [6], each hotspot has a sense range R for a specific error margin ΔT, which means if a sensor located within a Euclidean distance R from the hotspot, it can calibrate the temperature of the hotspot with an error less than ΔT. The equation is described in [6] as: $R = 0.5 \bullet K \bullet \ln(\frac{T_{\max}}{T_{\max} - \Delta T})$. From the equation we can see that the sensing range is decided by $T_{\max}$ of hotspots given that the error margin ΔT is predetermined. Since $T_{\max}$ is denoted as the maximum temperature difference, it can be calculate and recorded directly during simulation. And in the equation K stands for the thickness of the chip, which is also a predetermined parameter in our application.

Since each hotspot has its own value of $T_{\max}$, their sense range will also be different. So in step 1 we will not use the maximum sensing diameter. Instead we shall propose a specific function, so that given a cluster of points and a location of sensor, after adding an extra point, this function will find the new location of sensor and make the decision whether to add the point or not. Essentially the QT clustering algorithm does not require specifying the number of clusters a prerequisite, and always returns the same result when run several times. However, it requires more computing power than *k*-mean. Thus an efficient and time-saving method is needed for large data operation.

4.2.2 Find the Location of Sensors

4.2.2.1 Redundancy Distance

As the QT clustering algorithm runs, each time we add a point to the cluster, the location of the sensor for this cluster changes correspondently. Hence we need a criterion to determine the point to be added and use this to form a function to decide the new location of the sensor. For this purpose, we first introduce the definition of ***redundancy distance*** X between hotspot h and current sensor location S as:

$$X_{hs} = R_h - D_{hs} \qquad (8)$$

In equation(8) R_h stands for the sense radius of the hotspot and D_{hs} stands for the distance from h to s. If the sensor is within the sense range of h, the redundancy distance should be positive. While the sensor is outside the sense range, the redundancy distance becomes negative. To get a local optimal result, our goal should be to keep as many points with a positive redundancy distance in the cluster as possible. Thus choosing the points with the largest redundancy distance and relocating the sensor so that the minimum redundancy distance of the points in the cluster will be maximized should be a reasonable approach.

So we need to find a method that should have the property that the minimum redundancy distance will be maximized after relocation. We use equation(9) to describe the method we used to divide the hotspots and allocate them into different clusters :

$$X_{hs'} = \max_{\forall s_1 \in line(h,s)} \{\min_{\forall h_1 \in c} \{X_{h_1 s_1}\}\} \qquad (9)$$

In equation(9) h1 denotes one of the current hotspots in the cluster c, s1 is the points among $line(hs)$, and s' is the new location of the sensor. The redundancy distance X has a

property that if we move the new location of sensor along the line of the coming hotspot h and current sensor s as shown in figure14, we can find the location q that gives the optimal results.

We assume that the current cluster and the location of sensor are already optimized. Then each time we add a new point h' to the cluster, we assume that the sensor will perform one of the 3 actions: 1, pushed away from h' along the line connecting h' and s. 2, pulled towards h' along the line. 3, remain in the same location. No matter what actions it takes, the ultimate goal is to maximize the minimum X and ensure that the redundancy distance of all the points in the cluster is positive. In the next section we will show that along the line there will be such an optimal location and we will use a divide and conquer style approach to implement the algorithm and dramatically reduce the running time.

4.2.2.2 Proof of the Existence of Optimal Location

From empirical observation we can conclude that:

Lemma 1: If we move the new location of sensor along the line of h' and s, the minimum redundancy distance $X_{\min}$ is first increasing then decreasing or just decreasing.

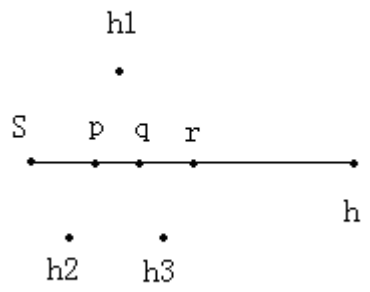


Figure 14. Illustration of the property of the redundancy distance

Here we shall give the proof for lemma 1 first:

1, It is quite obvious that if we move along any line that start from a point within a circle r, the redundancy distance X_{rp}, while p is any point in the line, will decrease after a period of increasing or just decrease.

2, As illustrated in figure14, we assume first that we move the sensor S towards the new hotspot h. Then we randomly pick 3 points, p, q and r respectively, from the line. And we assume that if we locate the sensor at point p, q or r, the minimum redundancy distance is achieved at hotspot h1, h2, and h3 respectively. We will prove our claim from 2 perspectives:

1), if $X_{h1p} > X_{h2q}$, for any point r beyond q, we have $\forall h \in S, \min\{X_{hr}\} < X_{h2q}$

Proof: Assume that there is a point r' in the line so that $\exists h' \in S, \min\{X_{hr'}\} = X_{h'r'} > X_{h2q}$. Since $X_{h1p} > X_{h2q}$, and $X_{h2p} > X_{h1p}$, we have $X_{h2p} > X_{h2q}$. Thus according to 1, for any point r beyond q, the redundancy distance between r and q satisfies $X_{h2q} > X_{h2r}$, which contradicts the condition that $\exists h' \in S, \min\{X_{hr'}\} = X_{h'r'} > X_{h2q}$. So statement 1) is proved.

2), if $X_{h2q} < X_{h3r}$, for any point p before q, we can say that $\forall h \in S, \min\{X_{hp}\} < X_{h2q}$

Proof: Assume that point p' in the line satisfy that $\exists h' \in S, \min\{X_{hp'}\} = X_{h'p'} > X_{h2q}$. Then in the same way we can find that $X_{h'p'} < X_{h2p'} < X_{h2q}$, which implies that the assumption will cause contradiction to the lemma 1. Thus we proved statement 2).

Above analysis is under the situation that the sensor is pulled towards the hotspot, but the same reasoning can be applied and consequently the same result will be obtained in the case that the sensor is pushed away from the hotspot. The proof gives us an idea that for a cluster of points and a sensor location, if we add a new point and move the new sensor location along the line connecting the current sensor location and the new hotspot, there will be a point p in the line that the minimum redundancy distance will be

maximized. And if we move the new sensor location along line [p, s] and line [p, h], the minimum redundancy distance will both be decreased. Hence Lemma 1 is proved.

4.2.2.3 Procedure for Location Determination

Utilizing the property described in Lemma 1, we can find the optimal value within a certain resolution using a divide and conquer style approach. The function can be described as:

STEP1: Assume that we have a cluster C and a sensor location S, and a new hotspot h is added.

STEP2: In the section between S and h, we equally choose n points, and calculate the minimum redundancy distance for each point.

STEP3: Choose the point with largest minimum redundancy distance X. For example, we assume that it is the ith point. Then from Lemma 1, we can see that the optimal location along the line lies between $(i-1)th$ and $(i+1)th$ point.

STEP4: perform the same procedure in the section between the $(i-1)th$ and $(i+1)th$ point, repeat until we get the desirable resolution

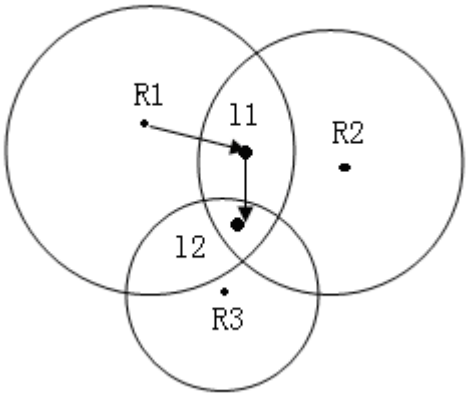


Figure 15. Find the location of sensor with our algorithm

Using such a divide and conquer style approach, we can reduce the computation time from O(k) to $O(\ln k)$, which is quite a significant improvement especially when a high resolution is needed. In figure15 is an illustration of the proposed algorithm. First we choose R1 as the starting point and the sensor location is also in the point R1. Then the second point R2 is added and the sensor location moves to l1. Finally R3 is added, and correspondently the sensor is relocated to l2.

4.2.3 Further Optimization

For the QT clustering algorithm, in most cases we do not need to consider all the possible cluster start point and just a portion of the points are enough. According to [13], the resulting solutions of the two approaches are almost the same at a considerable speed increase. And even if the results are different, there is no guarantee that the considering-all-points solution will turn out to be the better one. So based on the observation, we could just pick some of the points, say, in the order of sensing radius, and still get the optimized result we want.

Moreover, if we use equation(7) to obtain the thermal radii of the hotspots, it is very likely that we would not get a very accurate estimation. Thus we need a mechanism to modify this factor so that we can ensure that the final result will be optimized even if the initial guess of the sensing radius is not very accurate. The mechanism should include two variables: the hotspot's temperature precision error and its current sensing radius. If a certain hotspot's calibrated temperature has a relatively large error, we should decrease its radius accordingly. And the current radius will be used to determine how far the radius should be decreased.

4.2.4 The Flow of the Algorithm

The flow of the algorithm is shown is figure16:

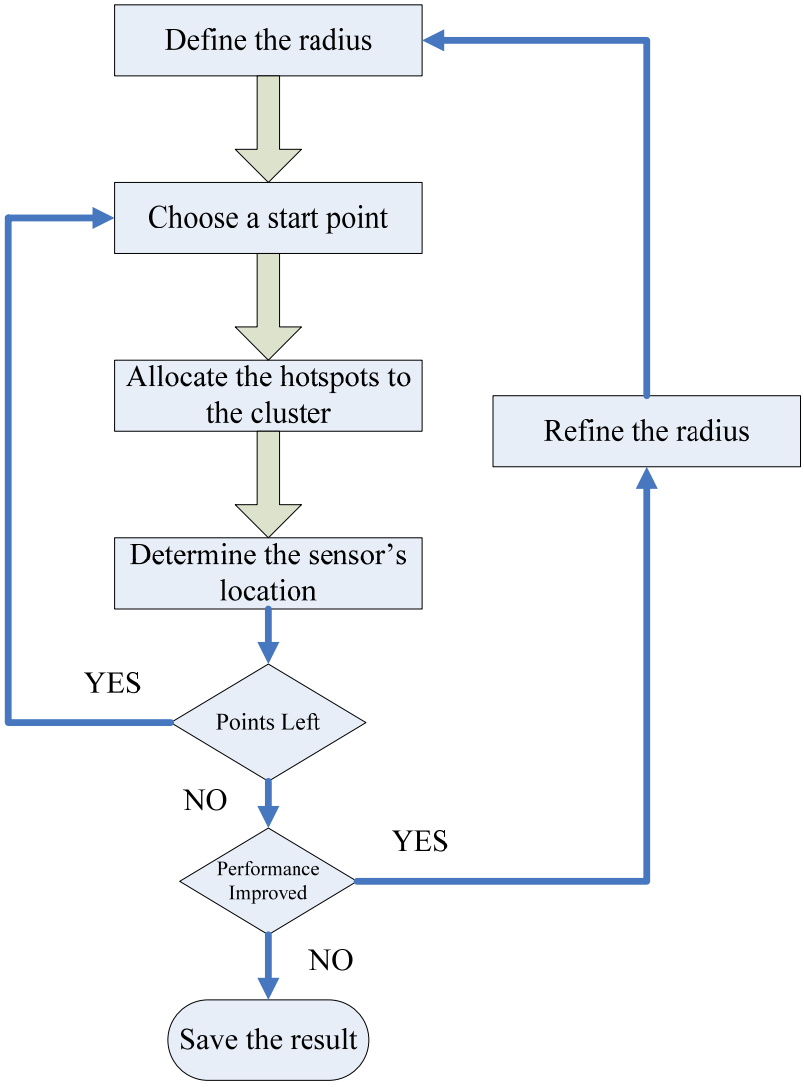


Figure 16. The flowchart of the QT clustering Algorithm

The flowchart indicates the complete flow about how the algorithm will work. However, since the radii refine method is not finished designing yet, we will not use this function in our experiment described in the next chapter and we will try to make sure that the sensor radius is as accurate as possible here.

4.3 QT Clustering Algorithm Experiment Result

4.3.1 Experiment Setup

To verify our algorithm, the first step is experiment setup. We use SIMPLESCALER[14] to simulate the behavior of SUPERSCLER processor with out-of-order issue and execution. The base processor we choose is Alpha 21364 as shown in figure16, and the default SPEC benchmark is run on the simulated processor. Then we use WATTCH infrastructure [15], which is an architecture level power modeling tool, to calculate the power distribution among the blocks. The generated power map will be fed into Hotspot [16] to generate the thermal map. Thermal simulators calculate a structure's temperature distribution based on its geometry, dissipation pattern, and material parameters. Mathematically, simulators solve a differential equation representing the heat transfer for the given boundary conditions. The most commonly used thermal simulator tools are based on the finite-element method (FEM). The great advantage of FEM programs is their generality, which means that they can simulate structures with arbitrarily complex geometry. FEM tools demand huge amounts of computer time. HOTSPOT version 3.0 or higher is capable of performing simulation both on block level and grid level. We mainly use its grid style simulation function here. The whole processor is divided into a bunch of small grids and the temperature in each grid is then computed by HOTSPOT. During each iteration, temperature and the location of the hotspots will be recorded to generate the final thermal map. After we get the hotspots distribution information inside a certain block, a program written by C will be applied to calculate the position of the sensor according to different algorithms used. And the temperature of the grid where the sensor is resided is considered to be the temperature

read by the sensor. If we set the grid size small enough, the accuracy of the temperature read can be guaranteed. The temperatures read by all the sensors will be compared with the temperature of the correspondent hotspot in the block, and the closest one will be accepted as the reading from sensors.

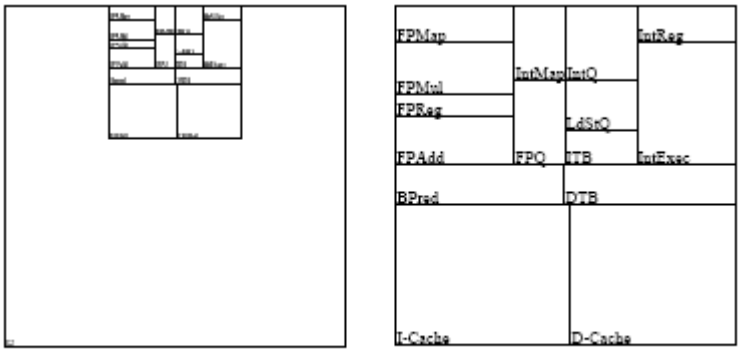


Figure 17. The image of processor alpha 21364.

4.3.2 Different Placement Algorithms

A Naïve Assignment

A naïve approach is to assign a sensor to the geometrical center of each block without considering the existence of the hotspots. And if we need to assign more than one sensor for each block, we just divide the block into sub-blocks and place the sensor in the center of the sub-blocks. For example, if we want to assign 2 sensors to a block, we just cut the block along the middle of the longer edge and divide the block in to 2 sub-blocks and place a sensor in each of them. We expect to have a worst result coming out for this algorithm.

2D K-mean Clustering Algorithm

In this algorithm we assign 2 sensors to each block and uses k-mean clustering algorithm to divide the hotspots into 2 groups. In each iteration, a hotspot is assigned to the cluster associated with a particular sensor if this sensor is the nearest one to it. And after all the hotspots being assigned, we simply use equation(14) to find the geometric center of the hotspots in the cluster to determine the new location of sensor.

$$O_{x,y} = \sum_{h \in M} (O_{x,y} + h_{x,y}) / size(M) \qquad (14)$$

In the equation $h_{x,y}$ denotes the coordinate of the hotspots and $size(M)$ is the number of hotspots included in the correspondent cluster M. No knowledge of temperature is used here, only the coordinates of the hotspots are needed. This process is repeated until the cluster divisions become stable and no further change can happen to the location of the sensors.

Modified 3D K-mean Algorithm

This algorithm is introduced in detail in CHAPTER2. We will follow the way it is described there to implement this algorithm and we will assign different number of sensors to different blocks. The bottom line is that the maximum precision error for all the blocks should not exceed 3 degrees.

QT Clustering Based Algorithm

The last algorithm we used for comparison is the QT Clustering Based Algorithm which is described in CHAPTER3. The maximum precision error ΔT we set for this test is 3 degrees and we will derive the radius of the hotpots accordingly. However, if we want to use equation(7) to find the range, we must notice here that this equation is just an

approximate estimation and the grid should be set larger enough, say, a few hundred square microns to make the estimation accurate. Thus without the self correction mechanism, we will refer to the actual thermal profile to make the estimation as accurate as possible.

4.3.3 Experiment Results

The result is shown in table1, we can see that the QT clustering algorithm finds the minimum sensor number distribution with only 23 sensors while ensuring the maximum precision error is within 3 degrees compared to 32 needed for 2D k-mean clustering algorithm and 25 for the improved 3D k-mean clustering algorithm. And for the 2D k-mean clustering algorithm, although it requires the largest number of sensors, it still fails to meet the maximum error requirement. However, the 2D k-mean algorithm prevails in the average precision error section as expected thanks to the extra sensors it uses.

	Sensor Number	Max Error	Average Error
QT method	23	2.848	0.2899
2D k-mean	32	3.466	0.1292
3D k-mean	25	2.7	0.2462

TABLE I. EXPERIMENT RESULTS OF DIFFERENT ALGORITHMS

In figure18 is a picture of the distribution of the hotspots and sensors. The small red dots represent the hotspots and the green rectangles represent the correspondent sensors whose locations are determined by the QT clustering based algorithm. The radii of the red dots vary according to the temperature of the hotspots, and the higher the temperature, the larger the radius is.

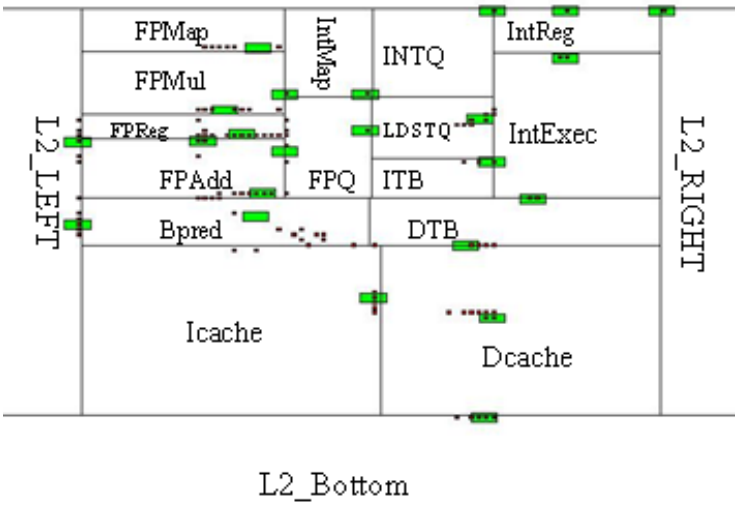


Figure 18. Sensor and hotspots distribution of QT clustering algorithm.

The figure shows a local distribution of sensors and hotspots. The algorithm runs on each block to determine the location of the sensor that monitors the temperature of the correspondent block.

4.4 Sensor distribution in global hotspots

Local hotspots are very important for thermal optimization schemes. However, in many occasions we need to look at the hotspots distribution globally and make decisions at a higher level. Previously we only consider the highest temperature on each block as the hotspots, but if we take the global thermal gradient into consideration, this might not be sufficient. So in this section we will look into the global sensor distribution problem. And we consider not only the point which has the highest temperature on chip during each application as hotspot, but also the point with local peak temperature. And as a result, the number of global hotspots increases from 25 to 56, as shown in figure 19.

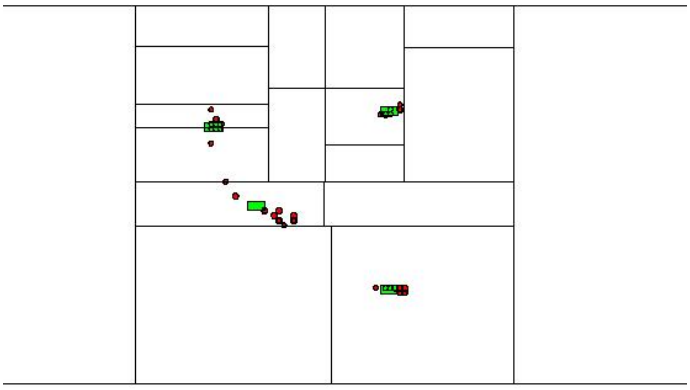


Figure 19. Global hotspots and sensors distribution.

The QT clustering algorithm is used to find the global sensor location. And in the final output, a minimum number of 4 sensors are needed for totally 56 hotspots, and the max precision error is 1.927 degrees and average precision error is 0.278 degree. The result is well optimized and generates a least number sensor distribution as expected.

4.5 Anisotropic Thermal Distribution

4.4.1 Introduction to Anisotropic Issue

In previous work, we assume that the thermal gradient around a certain hotspot is uniform and hence we can represent it with a circle. However, in real cases, this assumption usually is not valid and we should modify our algorithm to make it more precise and practical.

At the circuit level, thermal effects have important implications for both performance and reliability of the chip [36]. In general, on-chip temperature rise is known to increase transistor and interconnect delays and thereby degrade circuit

performance. The chip temperature rise is also known to degrade IC reliability since most reliability mechanisms have strong temperature dependence. In particular, interconnect performance and reliability factors are known to degrade with an increase in metal temperature. The interconnect reliability degradation is mainly due to the electro-migration (EM) phenomenon [7], although Joule heating (self-heating) in global lines is an additional contributor to the interconnect reliability degradation . The interconnect performance degradation is primarily due to the fact that the resistivity of metal interconnect increases linearly with its temperature, thereby increasing interconnect resistance (and consequently the signal propagation delay). In addition, the increase in the resistivity of the interconnect leads to an increase in interconnect Joule heating that causes an additional temperature rise.

The existence of thermal gradients on the substrate results in non-uniform temperature profiles along the length of these global interconnect lines running above the substrate, which in turn leads to non-uniform resistance profiles for these interconnect lines.

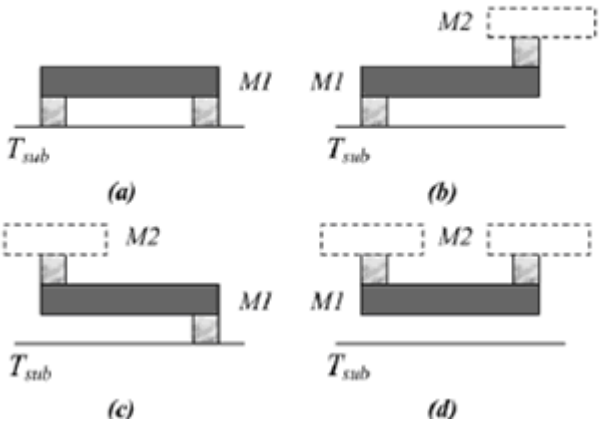


Figure 20. Different configuration of metal lines between M1 and M2.[36]

For example, as illustrated in figure20, the different configuration of metal line will certainly affect the thermal distribution of the substrate since most of the heat is dissipated through metal lines. And different via connection type will also influent the thermal flow. As a result, in practice the thermal gradient will not be uniform and we need take that into consideration in our algorithm.

4.5.2 Our solution

Since the thermal distribution is not uniform along the vertical and horizontal direction, the assumption that the sensing range around a certain hotspot is circular is not valid for most of the cases. In order to describe the temperature gradient around a hotspot as accurate as possible, an ellipse sensor range should be applied to the current algorithm.

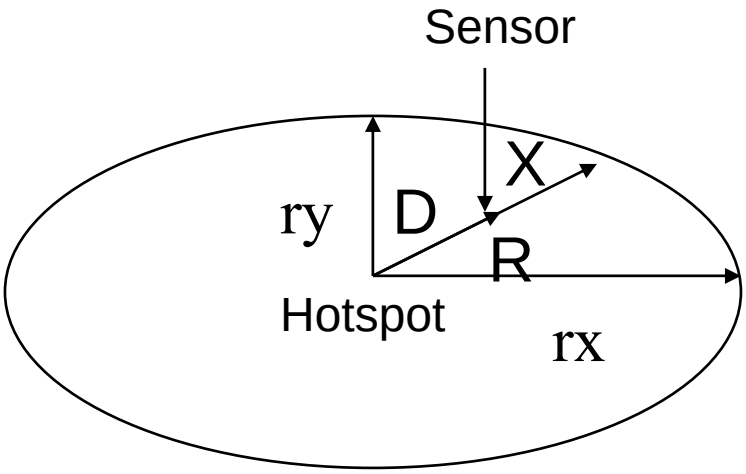


Figure 21. Modified sensing range with an ellipse shape

In figure 21 is an example of the modified sensing range. The hotspot is located in the center of the ellipse. Here D stands for the distance from the hotspot to the correspondent sensor location and R is the distance from the hotspot through the sensor

to the edge of the ellipse. And X remains as the redundancy distance that will be used in the algorithm. Since the shape of the sensing range has changed from circle to ellipse, we use rx and ry to represent the length in the X and Y axis respectively.

The main algorithm should be the same, while some modifications are still needed to suit the new sensing range. The new equation should be:

- Equation for ellipse: $\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} = 1$ (10)

- Equation for R: $R^2 = [1 + (\frac{h_y - s_y}{h_x - s_x})^2] * \frac{r_x^2 * r_y^2}{r_y^2 + r_x^2 (\frac{h_y - s_y}{h_x - s_x})^2}$ (11)

- The redundancy distance: $X = R - D$

In the equations, (h_x , h_y) is the coordinate of the hotspot and (s_x , s_y) is the coordinate of the sensor. So with the other coefficients remaining the same as we used in the original version of QT clustering sensor placement algorithm, we shall use the improved algorithm to determine the sensor location. In the resulted distribution, the sensor number seems to be the same as the original algorithm while having some improvement in the performance. It is quite reasonable since if we assign $r_x = r_y = R$, then the two algorithms is essentially the same.

4.5.3 Result and Conclusion

Results of the new sensor placement algorithm for each block are listed in table2 and in figure 22 are the comparison results with the original version of the algorithm.

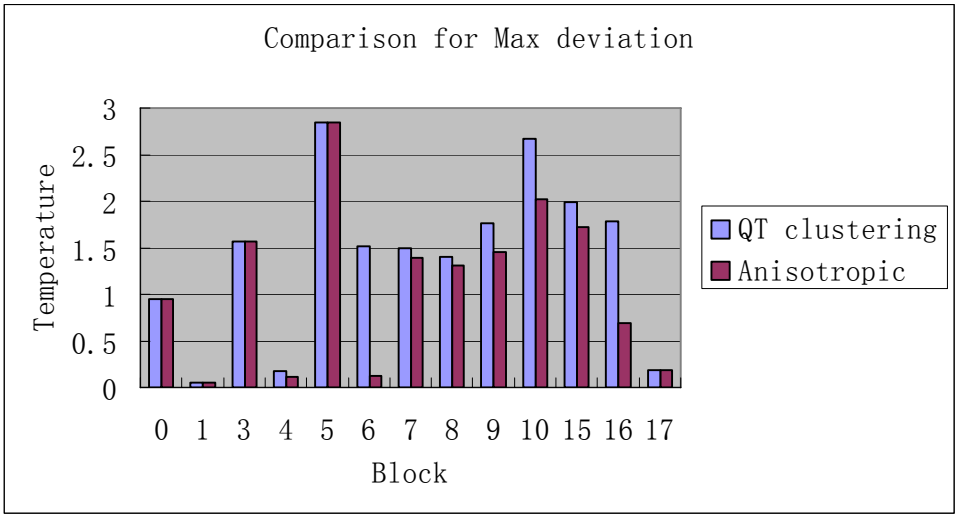


Figure 22. Comparison of maximum precision error.

Block	Anisotropic	QT cluster
0	0.952	0.952
1	0.051	0.051
2	0	0
3	1.569	1.569
4	0.117	0.176
5	2.848	2.848
6	0.123	1.515
7	1.396	1.496
8	1.314	1.401
9	1.453	1.759
10	2.024	2.665
11	0	0
12	0	0
13	0	0
14	0.077	0.077
15	1.721	1.987
16	0.686	1.785
17	0.184	0.184

TABLE II. MNOC SCHEME COMPARISON RESULT.

From the result we can see that the performance of the algorithm is enhanced by the new algorithm. However, the advantage and disadvantage of the anisotropic approach are both quite significant.

Advantage:

1. Better represent the real case on chip
2. Improve performance of the algorithm
3. Better suited for the border hotspots

Disadvantage:

1. Do not have a transform equation such as Lee's Equation to obtain the value of the coefficients directly from temperature.
2. Now we have two coefficients to optimize, which increases the complexity of the problem and time needed

We need to be careful and choose the proper optimization scheme in real implementation. In most cases the original algorithm is sufficient, while if accuracy is of great importance, the anisotropic-aware algorithm might be more proper.

CHAPTER 5

THERMAL SENSOR DISTRIBUTION IN MNOC

5.1 Introduction of MNoC

Complexity of VLSI circuits in terms of logic transistors that can be integrated on a chip is increasing at the rate of 58% per annum compounded. Whereas the design productivity growth is increasing at 21% per annum. As the technology advances, more and more different type of monitors, i.e. temperature, power, clock, are required. The data transmission between the controller and those many sensors poses a heavy burden to the bandwidth of the traditional buses. The cornerstone of the proposed platform is a concept that we call as *Monitor Network on a Chip or MNoC*. MNoC will be composed of computing resources in the form of processor cores and field programmable logic blocks, distributed storage resources, programmable I/O and all these resources interconnected by a switching fabric, allowing any chip resource to communicate with any other chip resource. Moreover, providing this intra chip communication based on switches will cause little or no overhead in the billion transistor scenario.

The integration of monitors and the collection and processing of monitor information is still an important unaddressed SoC design issue. As an initial step in the development of a complete monitor subsystem for SoCs, a low-overhead on-chip interconnect which is optimized for monitors, has been designed. This communication infrastructure allows for the interconnection of monitors ranging from complex performance analyzers to simple voltage sensors. Although simplified compared to other on-chip interconnect approaches, new interconnect technique supports irregular routing topologies, priority based data transfer and customized monitor interfacing. Collected

monitor data values are manipulated by one or more processors and the results are used to control SoC run-time operation.

The overhead and performance of the monitor network-on-chip interconnect for an eight core multiprocessor has been measured via hardware synthesis, interconnect simulation, and multi-core architectural simulation. For an eight core system, an area and power overhead of about 5% allows for the interconnections of 192 thermal monitors. The availability of a virtual channel for priority traffic results in the timely transfer of critical data. Architectural simulations show that multi-core performance can be improved by around 30% when MNoC-collected thermal data is used to perform dynamic frequency scaling.

Hence The MNoC concept has been necessitated by three factors: First there is an increasing demand of on-chip interconnect bandwidth. The second equally crucial factor is to amortize the enormous engineering cost involved in designing such large chips over multiple applications. The third factor is demand for easy-to-use methods to exploit the parallel processing capacity provided by multiple computational resources. Programmable interconnectivity and efficient implementation of shared memory abstraction and sensor resources are keys to provide this generality.

5.2 Previous Work

A lot of work has been contributed to the NoC problem. Although MNoC is not completely the same as NoC in application, it is still possible to use some conception of NoC to implement the MNoC architecture. For example, honeycomb type architecture is introduced[34] as shown in figure 23. The resources - computational, storage and I/O -

are organized as nodes of the hexagon with a local switch at the centre that interconnects these resources. Hexagons at the periphery would be primarily for I/O, whereas the ones in the core would have storage and computational resource. Each resource, located on a hexagonal node being connected to three switches, can reach 12 resources with a single hop. To further improve the connectivity, switches are directly connected to their next nearest neighbors, allowing any resource to reach 27 additional resources with two hops. As a last measure to further improve connectivity, every alternate switch is directly connected making each resource element reach a lot more elements with minimal number of hops.

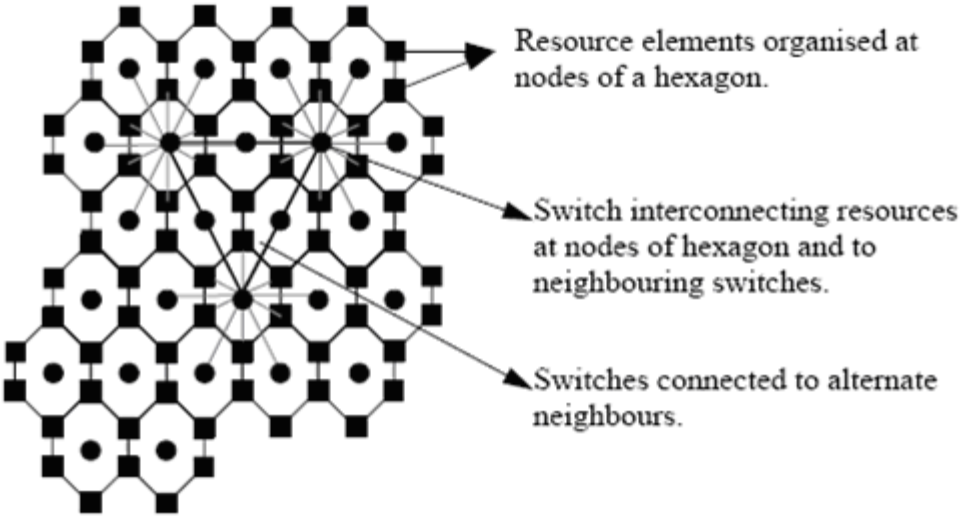


Figure 23. A honeycomb structure for NoC [34]

5.3, MNoC with Thermal Sensor Consideration

Most of the current explorations in the NoC area mainly focus on the power, regularity and cost issues. Specifically, in this chapter we will try to address the problem of the confliction of thermal sensor placement with the presence of MNoC architecture. The tradeoff here exists between the thermal sensor calibration accuracy and the

interconnect length, which will translate into the transmission delay, between sensors and the correspondent MNoC interfaces. Our ultimate goal is to seek a balanced and efficient thermal sensor distribution as well as a MNoC interfaces distribution.

A reasonable solution is to integrate the MNoC interfaces into the current thermal sensor placement algorithm. In that case, the choice of algorithm and the way the interfaces chipped in is crucial. For example, the QT clustering algorithm is not suitable to serve our purpose. There are several reasons: First of all, by the nature of the algorithm, once the sensing range of a certain hotspot is determined (here we treat the MNoC interface as a special hotspot), it requires at least one sensor's location to be within the interface's sensing range, which may cause the algorithm returning some additional unnecessary sensors and degrade the performance of the algorithm. Moreover, it is very hard to find a proper radius for the interface. The sensing circle of the interface must not cover the overlapping area containing the sensor, which we currently have no way to predict its shape and covering areas. If the overlapping area is enclosed inside the interface's sensing range, QT clustering algorithm will have no effect. So for this problem, our choice will be using k-mean clustering algorithm to find the new sensor location.

We will consider two different architectures: MNoC with a regular structure and MNoC without a regular structure. For the first case, the interfaces are treated as special hotspots. Thus we need to assign it two values to make it fit into the k-mean algorithm, which are its location and its temperature. Since the MNoC structure is assumed to be regular, the location of the MNoC interface should be predetermined and fixed during the algorithm. And for the temperature, we will set it to be the same as the

temperature of the sensor, which is the average temperature of all the current hotspots in the cluster. Beside those required factors, an additional coefficient k will be applied to the algorithm to represent the influence of the MNoC interface. The equation for k is:

$$X_{sensor} = (X_{hotspot0} + X_{hotspot1} + ...X_{hotspotn} + k*(X_{interface} - X_{sensor}) + X_{sensor})/(n+1) \qquad (12)$$

Here we recursively refine the value k so that an optimal value can be found. The introduction of the MNoC interface as a special hotspot will decrease the wire length from the sensors to the interface, but as a tradeoff, it might also increase the calibrated precision error. A balance should be achieved between the two factors and we can use the following equation as a criterion to determine the optimality of the algorithm:

$$E = (w'/w)/(T'/T) \qquad (13)$$

In the equation, w' is the reduced length of the wire, w is the total wire length, T' is the increased maximum deviation temperature number and T is the absolute value of the maximum deviation.

However, we should be careful when treating the MNoC interface as a special hotspot. One of the undesirable consequences is that the calibration accuracy will be greatly reduced since even just a few hundred microns' dislocation from the original position will cause a quite large difference in temperature measurement. Moreover, the computation time will also be increased because of the introduction of k. And the main advantage of this approach is that it can reduce the length of the wire from the sensor to the interface while maintain the regularity of the MNoC structure. The increased wire lengths will be translated into increased delay in conveying the temperature to the central

controller that takes corrective action through interface. And a larger delay requires taking actions more conservatively.

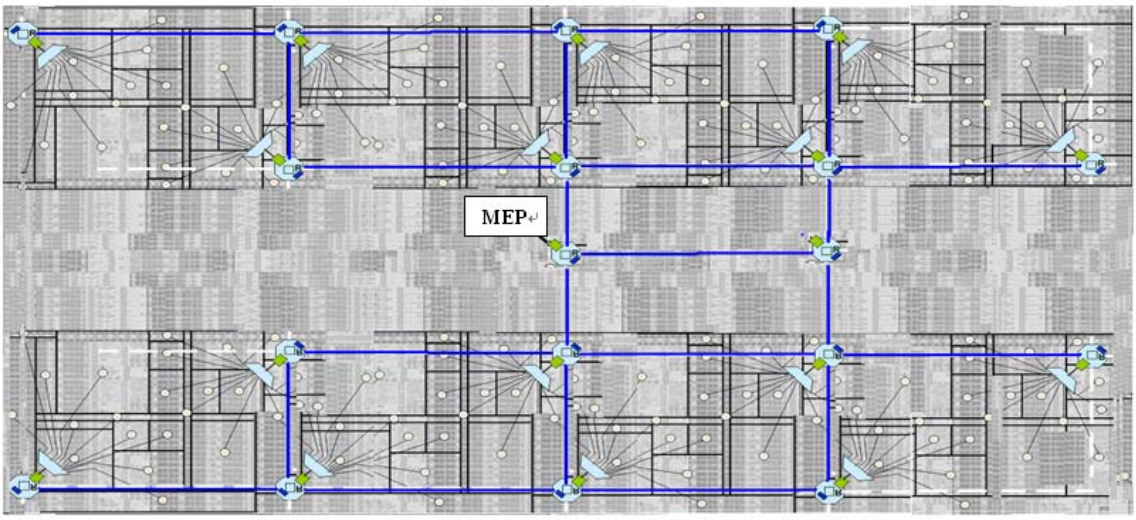


Figure 24. An example of interface distribution in a multi-core system

But if we do not need to maintain a very regular structure, then treating the sensor as hotspots and run the clustering algorithm for the interfaces which are treated as sensors can provide the best result. In that case we can significantly reduce the wire length while maintaining the calibration accuracy. That would be a very desirable result.

So two different approaches for various MNoC architectures will be implemented:

1, Do optimization with a regular MNoC structure. A k-mean clustering algorithm will be used to determine the sensor location while considering the MNoC interface as a special hotspot with a fixed location for optimization. And the final result will be the comparison result for wire-length and calibration resolution with the result returned by the original algorithm.

2, Do optimization with a flexible MNoC interface location within a certain range. And two different algorithms will be used to find both the sensors' and the interfaces' locations. First the proposed QT clustering algorithm will be used to determine the sensor location. Then we will use k-mean clustering algorithm to determine the MNoC interfaces' location. And the final output will be the sensors' and MNoC interfaces' distribution picture.

5.4 Experiment Validation

As we stated in the previous section, two different approaches are used to solve the thermal sensor distribution problem with the presence of different MNoC structures. The first implementation will be trying to find an optimal solution for the thermal sensor distribution problem under the assumption that there are two interfaces located in the fixed locations on chip. These two interfaces are located in the middle of each half chip(except for the L2_bottom cache since the thermal gradient is smooth and temperature is comparatively low there) respectively as shown in figure 24, with yellow rectangles representing the MNoC interfaces. And as stated in equation (12), different values of k are applied to find the optimal value. It is quite obvious that with a larger k, the influence of the interfaces will increase and thus we will have a greater chance to have an increased precision error and a reduced maximum distance from sensors to the interface.

k value	Max deviation	Max distance
0	1.927	0.011425
0.1	1.927	0.01113
0.2	2.821	0.011172
0.3	3.726	0.011045
0.4	5.443	0.010916
0.5	7.139	0.010789

TABLE III. RESULT FOR A REGULAR MNOC STRUCTURE.

Table3 shows the test results with k change from 0 to 0.5. When k=0, the location of the interfaces will have no influence on the determination of the sensors' locations. And in this case the algorithm reduced to the original 3d k-mean clustering algorithm as illustrated in Chapter 3.

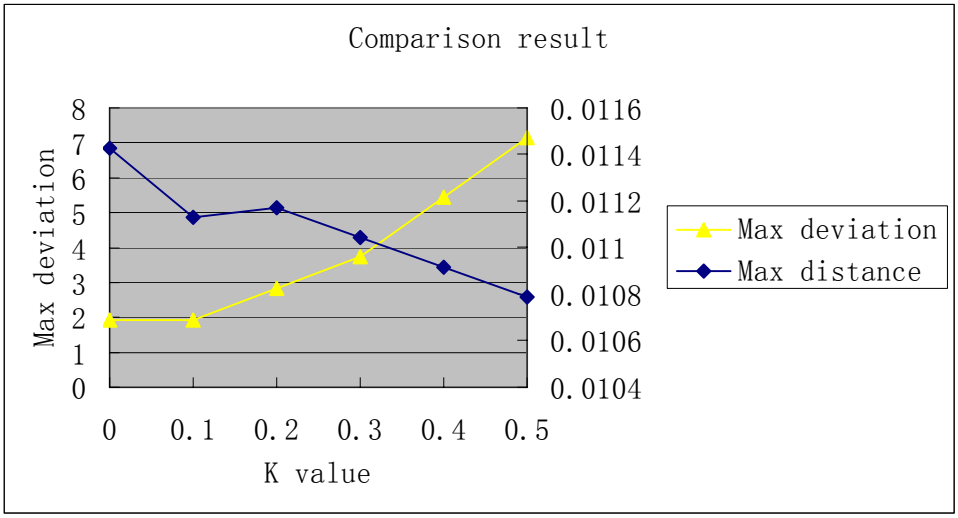


Figure 25. The maximum precision error and distance from sensor to interfaces.

In figure 25, it shows the maximum precision error and maximum distance from sensors to their correspondent interfaces when k varies from 0 to 0.5. In general the trend complies with our analysis. However, there are still some exceptions such as the maximum distance when k=0.2 compared to the maximum distance when k=0.1. The reason is because what we record here is the maximum distance and though the increased

coefficient k can guarantee an increase in the average distance, the maximum distance might not be strictly increasing as well.

And for the situation that we do not require a regular interface distribution, the solution will be much simpler. Since we decide the location of sensors using QT clustering algorithm on the hotspots, the maximum deviation should be the same as we derived in Chapter 4. And then we treat the sensors as hotspots and the interfaces as sensors and run algorithm again. Since our purpose is to find an optimal distribution with a fixed number of sensors, k-mean clustering algorithm is more suitable here.

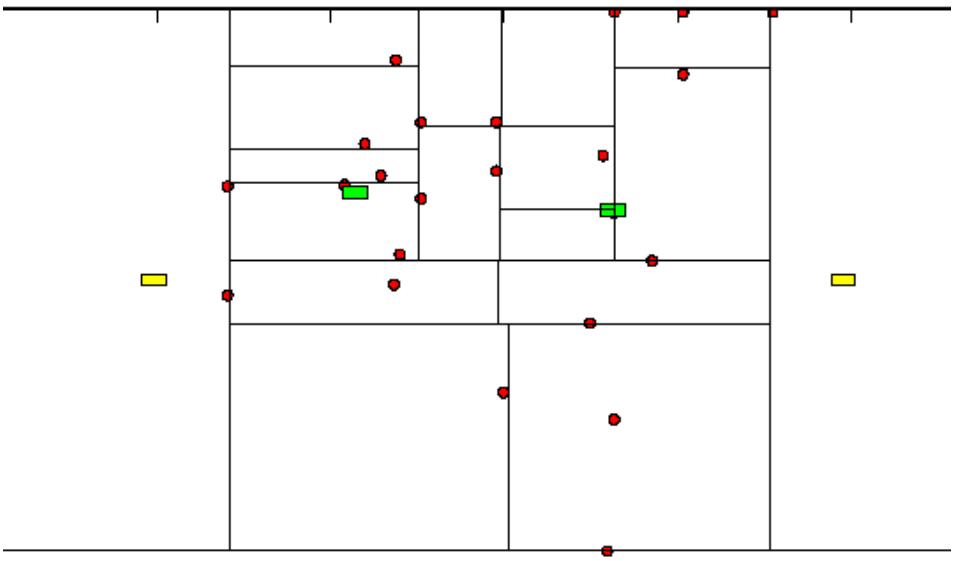


Figure 26. Sensors' and interfaces' distribution with an irregular structure.

The result is shown in figure 26. The yellow rectangles represent the regular MNoC interface which resides at the center of half chip. Green rectangles represent the optimized irregular interfaces. And the red dots are at the location of the sensors.

According to the nature of the algorithm, the maximum precision error should be the same as the original algorithm in the case of irregular interface placement. And we can see that the maximum distance has reduced from 0.0043271 to 0.003938, which means we have an improvement of about 10%. That is the case of the maximum deviation, and if we take the average precision error into consideration, the improvement will be even more significant.

CHAPTER 6

CONCLUSIONS AND PROPOSED WORK

We have introduced a new algorithm for systematic thermal sensor placement problem in microprocessors. The effectiveness of the algorithm depends on the accurate estimation of the thermal gradient of hotspots. We have proved that with an accurate calibration of the thermal gradient, the algorithm will outperform other algorithms quite significantly. The main advantage of the algorithm is that it can find the minimum number sensor distribution given a certain precision error limitation. We also take the non-uniform thermal gradient distribution into consideration and improved our algorithm based on the anisotropic property.

After finishing the fundamental algorithm design, we continue to implement the sensor placement algorithm combined with other techniques such as monitor network on chip (MNoC). We tried to implement the algorithm under different constraints and the comparison results show that the proposed scheme can improve the over-all performance quite significantly in terms of wire-length and temperature calibration accuracy.

For the future work, one possible application is to implement the algorithm in 3D structures. However, there are also some problems in 3d structure implementation. For example, since the heat dissipation between 3d layers flows mainly through metal lines(via), the thermal gradient will not be a sphere as we used in the model and hence the sensing range becomes more unpredictable. That will increase the complexity of the problem. Moreover, hybrid sensor distribution, such as combination of the traditional thermal sensors and interconnect thermal sensors, is also a possible area for exploration.

BIBLIOGRAPHY

[1] V. Krinitsin, 'Pentium4 and Athlon XP: Thermal Conditions', http://www.digit-life.com/articles/pentium4athlonxpthermalmanagement/

[2] K.Skadron, M.R.Stan, K.Sankaranarayanan, W.Huang, 'Temperature aware Microarchitecture', In Proc. ISCA, 2003

[3] R Mukherjee, SO Memik, 'Systematic temperature sensor allocation and placement for microprocessors', In Proc. DAC, 2006

[4] M. Gomaa, M.D. Powell, T.N. Vijaykumar, 'Heat and Run: Leveraging SMT and CMP to manage power density through the operating system', In Proc. ASPLOS, 2004

[5] S. Heo, K. Barr, K. Asanović, 'Reducing power density through activity migration', ISLPED, 2003

[6] K.J. Lee, K. Skadron, W. Huang, 'Analytical model for sensor placement in microprocessors', ICCD, 2005.

[7] S. Gunther, F. Binns, D.M. Carmean, J.C. Hall, 'Managing the impact of increasing microprocessor power consumption', Intel Technology Journal, Feb 2001.

[8] P. Bratek, A. Kos, 'Temperature sensor placement strategy for fault diagnosis in integrated circuits', In Proc. SEMI-THERM, 2001

[9] R. Mukherjee, S. Mondal, S.O. Memik, 'Thermal sensor allocation and placement for reconfigurable systems', ICCAD, 2006.

[10] S. Mondal, R. Mukherjee, S.O. Memik, 'Fine-grain thermal profiling and sensor insertion for FPGAs', In Proc. ISCAS 2006.

[11] S. Lopez-Buedo, J. Garrido, E.I. Boemo, 'Dynamically inserting, operating and eliminating thermal sensors of FPGA-based systems', IEEE Transactions on components and packaging technologies, Vol.25, No.4, Dec 2002.

[12] QT Clustering Algoritm drscription http://en.wikipedia.org/wiki/Cluster_analysis.

[13] QT Clustering property http://rss.acs.unt.edu/Rdoc/library/flexclust/html/qtclust.html.

[14] Burger, D.C. and T.M. Austin, *The SimpleScalar Tool Set, Version 2.0.* Computer Architecture News, 1997. **25**(3): p. 13-25.

[15] Brooks, D., et al. *Wattch: A Framework for Architectural-Level Power Analysis and Optimizations*. in *ISCA*. 2000.

[16] Skadron, K., et al. *Temperature-Aware Microarchitecture*. in *ISCA*. 2003.

[17] P. E. Gronowski, W.J. Bowhill, R. Preston, M. Gowan and R. Allmon " High Performance Microprocessor Design" IEEE Journal of Solid State Circuits, vol.33, no.5, pp 676-686, May 1998 .

[18] S. Borkar, T. Karnik, S. Narendra, J. Tscahnz, A. Keshavarzi and V.De, "Parameter variation and impact on circuits and microarchitecture", Proceedings of Design and Automation Conference (DAC), 2003, pp. 338-342.

[19] Ajami, A.H.; Bnerjee, K.; Pedram, M.; van Ginneken, L.P.P.P, "Analysis of non-uniform temperature-dependent interconnect performance in high performance ICs" Design Automation Conference, 2001. Proceedings Volume , Issue , 2001 Page(s): 567 - 572.

[20] W.Hung, C.Addo-Quaye, T.Theocharides, Y.Xie, N.Vijaykrishnan and M.J.Irwin, "Thermal-Aware IP Virtualization and Placement for Networks-on-Chip Architecture," 2004 IEEE International Conference on Computer Design (ICCD'04) pp. 430-437.

[21] Y.Xie and W.Hung, "Temperature-Aware Task Allocation and Scheduling for Embedded Multiprocessor Systems-on-Chip (MPSoC) Design", The Journal of VLSI Signal Processing 2006, pp. 177-189.

[22] J.W.Haskins and K.Skadron, "Minimal Subset Evaluation : Rapid Warm-Up for Simulated," 2001 IEEE International Conference on Computer Design (ICCD'01) p. 0032.

[23] R.Mukherjee, S.Mondal and S.O.Memik, "A Sensor Distribution Algorithm for FPGAs with Minimal Dynamic Reconfiguration Overhead," international conference on Engineering of Reconfigurable Systems and Algorithms 2006.

[24] T.C.Shermer "Recent results in Art Galleries" PROCEEDINGS OF THE IEEE, VOL. 80, NO. 9, SEPTEMBER 1992

[25] L.Buedo and E. Boemo."Making Visible the Thermal Behaviour of Embedded Microprocessors on FPGAs. A Progress Report". . FPGA'04, February 22–24, 2004, Monterey, California, USA

[26] V.S.M.Rencz and B.Courtois, "Tracing the Thermal Behavior of ICs",IEEE DESIGN&TEST of COMPUTERS 1998

[27] S.S.Dhillon and K.Chakrabarty,"Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks", *Proc. IEEE Wireless Communications and Networking Conference*, pp. 1609-1614, 2003

[28] D. Estrin, R. Govindan ,J. Heidemann, S. Kumar,"Next Century Challenges: Scalable Coordination in Sensor Network", Mobicom '99 Seattle Washington USA

[29] D. GANESAN, R. CRISTESCU, and B. BEFERULL-LOZANO,"Power-Efficient Sensor Placement and Transmission Structure for Data Gathering under Distortion Constraints" ACM Transactions on Sensor Networks, Vol. 2, No. 2, May 2006, Pages 155–181

[30] Frank Y. S. Lin and P. L. Chiu,"A Near-Optimal Sensor Placement Algorithm to Achieve Complete Coverage/Discrimination in Sensor Networks",IEEE COMMUNICATIONS LETTERS, VOL. 9, NO. 1, JANUARY 2005

[31] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, " Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks", IEEE TRANSACTIONS ON COMPUTERS, VOL. 51, NO. 12, DECEMBER 2002.

[32] K.Lee and K. Skadron, " Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors", Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)

[33] A Hemani, A Jantsch, S Kumar," **Network on a Chip: An architecture for billion transistor era**", Proceeding of the IEEE NorChip Conference, November, 2000

[34] S.Kumar, A.Jantsch, J.Soinninen," A Network on Chip Architecture and Design Methodology", Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI.02)

[35] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho," Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks", IEEE TRANSACTIONS ON COMPUTERS, VOL. 51, NO. 12, DECEMBER 2002.

[36] A. H. Ajami, K. Banerjee, and M. Pedram,"Modeling and Analysis of Nonuniform Substrate Temperature Effects on Global ULSI Interconnects", IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 24, NO. 6, JUNE 2005

[37] R. Vadlamani, Wayne Burleson and Russell Tessier, "Employing a Monitor Network-on-Chip in a Shared Memory Multiprocessor Rollback Recovery System "technical report for SRC task ID 1595