Thermal-Aware Scratchpad Memory Design and Allocation

Morteza Damavandpeyma[1], Sander Stuijk[1], Twan Basten[1,2], Marc Geilen[1], Henk Corporaal[1]
[1]*Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands*
[2]*Embedded Systems Institute, Eindhoven, The Netherlands*
{m.damavandpeyma, s.stuijk, a.a.basten, m.c.w.geilen, h.corporaal}@tue.nl

***Abstract* — Scratchpad memories (SPMs) have become a promising on-chip storage solution for embedded systems from an energy, performance and predictability perspective. The thermal behavior of these types of memories has not been considered in detail. This thermal behavior plays an important role in the reliability of silicon devices and in their static (leakage) power consumption. In this paper, we propose two different techniques to improve the thermal behavior of SPMs. First, we propose a hardware-based, thermal-aware address translation technique that physically distributes memory accesses to consecutive addresses evenly over the whole memory area. Second, we propose a software-based, thermal-aware address generation technique. This technique tries to distribute the variables that are allocated to the SPM in such a way that an even thermal distribution is achieved. The first technique works particularly well for applications with a regular access pattern, whereas the second technique can also improve the behavior of applications with irregular access patterns. The two techniques thus complement each other and work well together. Using the first technique we show that the peak temperature of an SPM in 65nm technology, when running a typical streaming application, is decreased by up-to 10.0°C. Temperature cycling is reduced from up-to 14.8°C to almost zero in comparison with a non-thermal-aware solution. For our benchmark applications with an irregular access pattern, the second technique is able to reduce the peak temperature by up-to 3.5°C. These savings for both techniques are obtained without any performance degradation or extra silicon area.**

I. INTRODUCTION

Most applications in embedded systems perform complex processing operations on a stream of input data. These operations are typically power hungry. Low power/energy methods have been employed to increase the availability of handheld devices as much as possible. Studies have shown that memory systems are contributing to a large portion of the total energy consumption [1]. To increase performance and decrease the energy consumption, memory hierarchies are introduced to limit the number of off-chip memory accesses either by using caches, SPMs, or both. SPMs have become an efficient replacement for caches in novel embedded systems, due to their lower energy/area cost and better predictability [2]. It has also been demonstrated [3] that for applications with a regular data access pattern SPMs can give better memory behavior when compared to caches.

The drastic increase of the power density of digital circuits by shrinking feature sizes of transistors has become an important concern in the VLSI industry. This aspect has especially a large impact on memories as their density is higher than other parts. Higher power density translates to a higher local chip temperature. In the deep submicron domain, leakage power has an exponential relation with temperature [4]. As a result of the higher power density, temperature and therefore leakage power will increase [5]. Cooling systems can be a way out of this issue, but the use of a large cooling system to lower this high temperature causes extra design cost and a larger product size. It has also been proven that temperature plays an important role in the reliability of silicon devices [6]. The mean time to failure (MTTF) of silicon devices, which depends exponentially on temperature [10], will be affected if this issue is not considered during design. Another side effect of a temperature rise is an increase in the interconnect delay [7] which is due to dependency of the resistance of the interconnect on the line temperature. This dependency directly affects the predictability of the device.

The number of accesses to the SPM will affect its power consumption and a change in the power consumption will change the thermal behavior of the SPM. So to improve the thermal behavior of an SPM, it is necessary to consider the access pattern of an application onto the SPM. We categorize applications into two groups: applications with a regular memory access pattern and applications with an irregular memory access pattern. We propose two techniques, each one specifically targeting one category. The two techniques complement each other and do not interfere. When put together, the techniques improve the thermal behavior for all applications.

Many streaming applications that are mapped onto embedded systems have a regular memory access pattern (e.g. motion estimation in an H.263 encoder). When using an SPM with a traditional address layout, consecutive elements in large variables (e.g. arrays) are mapped onto consecutive physical addresses in the SPM. Subsequent memory accesses of an application that has a regular memory accesses behavior will therefore be performed on memory locations that are close to each other. At each moment in time, there will be one region in the SPM that is heavily accessed. This will incur a high power density in this region. Over time, the highly accessed region will move over (part of) the SPM. This will lead to a moving hotspot on the SPM and as a result it leads to thermal cycling. This paper proposes a novel simple SPM address decoding strategy that takes the regular memory access pattern found in streaming applications into account. The use of the proposed address decoder leads to a balancing in the access pattern, power density, and temperature distribution over the physical area. Our simulation results show that by using our proposed technique, the peak temperature reduces by up-to 10.0°C in a motion estimation application and up-to 2.6°C in an image processing application when considering the accesses to the video frames and image blocks respectively. Temperature cycling decreased from 14.8°C to 0.1°C and from 3.8°C to 0.1°C, respectively.

For applications with an irregular memory access behavior, we propose a novel thermal aware address generation technique. The proposed technique creates a thermal equilibrium in the SPM by constructing thermal-

This work is supported in part by the Dutch Technology Foundation STW, project NEST – 10346.

aware addresses for the variables that are allocated to the SPM. Using our technique, we are able to reduce the peak temperature of an SPM that contains the most frequently accesses variables in MP3 decoder, by 3.5°C when compared to a traditional SPM allocation technique.

The remainder of this paper is organized as follows. Section II discusses related work on SPM management methods and thermal-aware design. The proposed technique for applications with a regular access pattern is discussed in Section III. Section IV introduces the proposed technique for applications with an irregular access pattern. Section V explains the experimental setup that is used to evaluate our techniques. Section VI shows the experimental results. Section VII concludes this paper.

II. RELATED WORK

SPMs have been studied extensively to find an optimal mapping of code or data from the performance and energy perspectives [11][12][13]. In this paper, we optimize SPMs from a thermal perspective. Thermal issues were studied before for different elements in microprocessors (e.g. cache memory [8], register files [9], etc) to improve their reliability and thermal behavior. In [8], a thermal-aware method is introduced to reduce hot spots inside a cache. The authors propose a block permutation scheme to maximize the physical distance between the highly accessed regions of the instruction cache. Inspired by [8], but adapted to SPMs, we apply a simple logical to physical address translation scheme to increase the physical distance between consecutive data accesses in an SPM with the objective to achieve an even thermal distribution in case of applications with a regular access patterns. In [14], a technique is employed to handle the thermal issues of systems with a cache and SPM, by swapping memory blocks between the cache and SPM at run-time. The aim is to reach an equilibrium temperature and an improvement in the reliability of the system. In [15], a block duplication scheme under compiler control was proposed. This technique is used to deal with soft-errors in SPMs and as such to increase the reliability of the system. We focus on improving thermal behavior and, as a result of that, improving leakage power and life-time reliability.

Our work is the first that considers thermal issues for a system with only SPMs. Compared to [14] which uses a combination of caches and SPMs to achieve a better thermal distribution by swapping data between SPM and cache, in our work, the SPM can work stand alone and there is no need for extra cooperation between the SPM and other elements. The techniques proposed in this paper can be applied to any SPM. So, in principle also on top of the technique proposed in [14]. Also, to the best of our knowledge, this paper is the first one that considers the effect of SPM address generation on the SPM's thermal behavior. The position of the variables inside the SPM will affect the thermal dissipation flow. Taking the effect on temperature into account during address generation provides a lower power density throughout the SPM, leading to a reduction in peak temperature.

III. THERMAL-AWARE SPM DESIGN FOR APPLICATIONS WITH A REGULAR ACCESS PATTERN

Our first aim is to enhance the thermal behavior of an SPM by reducing its peak temperature and by reducing the thermal cycling for applications with regular memory access patterns.

A. Observation

Multimedia applications typically process various types of audio or video data streams. These data streams are placed as a sequence of frames or macro-blocks in the memory system. Applications access this data in a regular manner. During a short time period (e.g. the processing time of one macro block of an image) one region of the memory (e.g. storing a macro block) is accessed heavily. As a result of this high activity, the power density of the accessed region increases and the local temperature rises. In multimedia applications, several consecutive macro-blocks are loaded into the SPM. During run-time, different physical regions of the SPM experience fluctuation in their temperature. Fig. 1 shows 16 different temperature maps of an SPM. (These maps are obtained by a simulation tool flow described in Sec. V.) In each temperature map, a different macro-block is accessed by a motion estimation application. Initially the application accesses macro block B0. The temperature map at the left-top of the figure shows the temperature map of the SPM at the end of the time period in which this block is accessed. Once this block is processed, the application continues with macro block B1. The temperature map of the SPM after processing this block is shown as the second map on the top line of the figure. After processing this block, the application continues with block B2 etcetera till it has processed the last block (i.e. B15). In each period of access in Fig. 1, one specific memory bank is accessed and as a result, its temperature becomes the highest of the whole SPM. From the reliability perspective, continual changes in the temperature of a silicon device will lead to fatigue in the device and finally device failure [21]. The simplest way to decrease the local power density is by distributing the accesses across the whole SPM area. In the next sub-sections, we demonstrate how we can create this equilibrium situation during the execution of an application.

B. Logical Address to Physical Location Translation

Because the subsequent memory accesses of an application are typically performed on subsequent logical addresses, an even distribution of accesses across the SPM can be achieved by maximizing the physical distance between consecutive logical memory addresses. To realize this objective, an encoding scheme is required to translate consecutive logical addresses into distributed physical locations. Fig. 2 illustrates this translation by means of a simple explanatory example. In this example, we assume that each bank contains four data elements (words). When translating the logical addresses into the physical address locations, we can increase the physical distance between these four words.

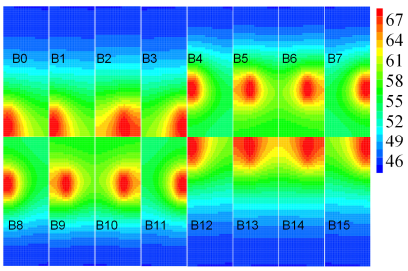

Fig. 1. Temperature of SPM while running the motion estimation application.

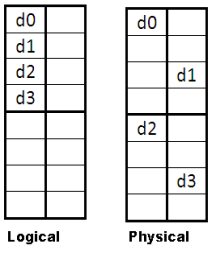

Fig. 2. Logical to physical address encoding

C. Implementation

It is possible to find different optimal solutions for the address translation problem described above. At the same time, we require that our encoding scheme (i.e. address decoder of the SPM) must be as simple as possible in order to limit its implementation complexity and to incur as little overhead on the system as possible. Therefore, we propose a heuristic to solve the encoding problem. Based on this heuristic, an encoding technique from the logical address space to the physical address space is derived. The suggested heuristic is not necessarily always optimal in terms of distributing accesses. The advantage of the proposed heuristic is that it has a simple implementation in hardware without extra hardware cost or performance overhead. Assume a memory consisting of C×R banks, with C and R the number of columns and the number of rows in the memory bank structure, respectively. Assume each bank has N words.

Heuristic. *Choose m=N/(C×R) words from each memory bank in order to place these m words with distance d=C×R from each other inside the memory. Repeat the previous action C×R times to choose all N words.*

A simple and efficient way to implement this heuristic is to change the address decoder of the SPM. The following illustrates the proposed address translation method:

Address mirroring. *The desired effect can be achieved via address decoding by just mirroring the address bits. Consider the following logical address:*

$$A_s, A_{s-1}, \ldots, A_3, A_2, A_1$$

The corresponding physical address is set to:

$$A_1, A_2, A_3, \ldots, A_{s-1}, A_s$$

where $s=\lceil \log_2(C\times R\times N)\rceil$.

By mirroring the logical addresses, consecutive accesses within one bank are distributed over all banks of the SPM with a distance of *d* words from each other. The address mirroring scheme explained above is inspired by [8]. In [8], an address permutation scheme was used to create a permutation of blocks inside an instruction cache memory. In a cache, it is reasonable to preserve the relation between the elements of each memory bank to keep locality of cache lines. In an SPM this is not necessary. Our mirroring scheme increases the physical distance between consecutive accesses to a memory bank and even increases the distance between the currently accessed memory bank and the previously accessed memory bank.

Fig. 3 shows a simple example of our address decoding scheme with a memory that consists of 4 banks and 16 words inside each bank (N=16, C=2, and R=2). The memory layout on the left hand side of the figure shows accesses to the SPM when a conventional address decoding is used and the right hand side shows the memory layout of the same memory accesses when the proposed address decoding is used. The figure shows that the proposed technique is able to increase the physical distance between consecutive SPM accesses. Note that the proposed solution requires only a change in the address lines of the SPM, so no extra hardware is required. In other words, no extra silicon is needed and no performance loss will occur when using the proposed technique.

IV. Thermal-aware SPM allocation for applications with an Irregular access pattern

In the previous section, we proposed a hardware-based technique to improve thermal behavior in an SPM for applications which have regular access patterns. In this section, we propose a software-based SPM allocation technique to optimize thermal behavior of the SPM in applications with irregular access patterns. In applications with irregular accesses to memory, different variables which are chosen for the SPM, have different types and sizes. Each variable has a different number of accesses during the execution of the application. This will lead to a different power consumption and different temperature for the different memory locations where these variables are stored. Better thermal dissipation can be achieved by placing the variables in a way that creates an even memory access density throughout the SPM. In the next subsections, first we describe our method to order variables based on their expected temperature and then we will propose an address generation technique for SPM variables to take into account their thermal behavior. We assume that the set of variables to be considered has been generated by an SPM allocation technique such as for example the one of [12].

A. Thermal Ordering for SPM Variables

Power dissipation directly translates to temperature in silicon devices. From the number of accesses to memory elements, the power consumption can be calculated. So it is possible to order SPM variables from the expected highest temperature (hot side) to the expected lowest temperature (cold side) if the number of accesses to each SPM variable is known. We use a profiling technique to estimate the number of accesses to each SPM variable. We instrument the application source code with profiling instructions to count read and write operations. After compiling and executing the instrumented source code, it is possible to separate the most frequently accessed variables from the less frequently accessed ones and give an ordered list of SPM variables. Our goal is to create an even access density to the SPM, equal to the average access count over variables. As the sizes of the variables may be different, this must be taken into consideration during the SPM allocation steps. We consider an average of the accesses to different elements of an array as the number of accesses to each element of the array. This assumption is valid in most applications (e.g. accesses to a frame buffer in multimedia applications) or near-optimal from the perspective of our objective. Current compilers place elements of an array in a consecutive order in memory. So a larger variable causes a higher power consumption. By multiplying the size of a variable with its average access count, we get an estimation of the amount of power that is dissipated through accesses to that variable. Fig. 4 shows an explanatory view of variable ordering based on the number of accesses and size of the variables. The product of the average number of accesses to each variable and its size is the weight of a variable. Based on these weights, we sort the variables from the highest weight (the hottest) to the lowest weight (the coldest).

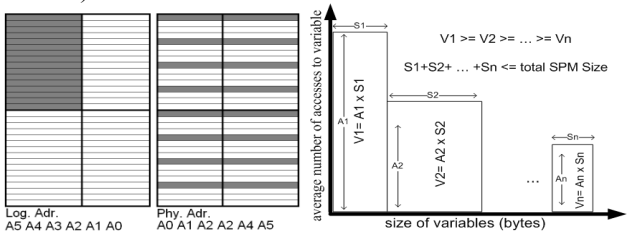

Fig. 3. Logical to physical address mapping.

Fig. 4. Ordering of SPM variables

B. Thermal Aware Address Generation for SPM

To get an even thermal distribution we put some cold variables between any two hot variables. As the size and average number of accesses to each variable is different, the number of cold variables that is placed between the hot variables is different throughout the SPM. The goal of the proposed technique is to place the variables in a way to create groups of one hot and some cold variables in such a way that the average of the assigned weights in a group is at most the average of the weights of all variables in the SPM. The HotColdEqualize (HCeq) algorithm, see below, starts with placing a hot variable; then it places some cold variables after the hot one such that the above condition on the average weight of the hot variable and the cold variables is satisfied. The algorithm stops if all variables in the list are placed in the SPM and received an address.

HotColdEqualize (HCeq)

```
adr =0
W = average weight over the variable list (VarList)
while VarList is not empty do {
    assign adr to VarList(1)
    adr = adr + size(VarList(1))
    L= length of VarList
    take N such that variables VarList(1) and VarList(L-N) … VarList(L)
        have an average weight ≤ W or N=L-1
    for i=1 to N {
        assign adr to VarList(L-i)
        adr = adr + size(VarList(L-i))
    }
    remove VarList(1) and VarList(L-N) … VarList(L) from VarList
}
```

In this paper, we apply HCeq along with a static SPM allocation technique. It is also possible to use the proposed address generation technique for dynamic SPM management. We leave the complete implementation of address generation for dynamic SPM as future work.

V. EXPERIMENTAL METHODOLOGY

This section explains how we combine the different tools that are used in our experimental setup.

A. Temperature Model

Due to the electrical-thermal duality, it is straightforward to use electrical concepts for modeling thermal behavior. The thermal model can be derived from the electrical model by replacing voltage with temperature, current with power, resistance by thermal resistance, capacitance by thermal capacitance, and electrical RC constant with thermal RC constant. In this work, we used HotSpot 5.0 [5] as our thermal simulation tool, which offers a platform to simulate thermal behavior. In our experiments to find the possible temperature range of an SPM during the execution of an application, we use two different scenarios. In the first scenario, we put the SPM as a single element in a die surrounded with air. In the second scenario, we put the SPM within a large inactive silicon area. The result of the first scenario shows the temperature distribution in the SPM when heat transfer with the surrounding is negligible (air is a good insulator). The second scenario shows the temperature distribution in an SPM when the SPM can transfer heat through the surrounding easily (silicon without power dissipation). Using these two scenarios, we can show the performance range of our proposed techniques in the best case and worst case situation based on the physical position of the SPM with respect to other elements (core, interconnects, etc) inside the chip.

B. Experimental Setup

An experimental setup was devised to evaluate the effect of the proposed method on the thermal behavior of an SPM. Fig. 5 shows the tool flow used in this paper. We used SimpleScalar [17] and Valgrind [18] to extract timing information and memory access patterns, respectively. SimpleScalar is a micro architectural simulator and Valgrind is a tool that can be used to trace the memory operations of an application. Table 1 shows the configuration of SimpleScalar that was used in our experiments. We added a patch to Valgrind to profile the access pattern of a user-selected set of variables inside an application. We used CACTI 6.0 [19] to determine the amount of dynamic power consumption of each access to the SPM, which is assumed to be implemented with an SRAM. Inputs to CACTI are the size of the SPM and its detailed configuration (e.g. size of memory banks (set to 256 bytes), number of read ports (set to 1), number of write ports (set to 1), and bus width (set to 32 bits)). The power trace is created by multiplying the access trace file from Valgrind by the amount of power consumed on each access. The latter number is obtained from CACTI. To determine the temperature of the SPM, we used HotSpot to perform a transient temperature simulation for the SPM layout. In our experiments, HotSpot was further configured as follows: the clock frequency is set to 1GHz, the ambient air temperature is set to 45°C, the initial silicon temperature is set to 45°C, and the rest of the parameters are left with their default setting.

VI. EXPERIMENTAL RESULTS

To evaluate our techniques we use two applications that have a regular memory access pattern and one application with a mostly irregular memory access behavior. The first application is motion estimation [16], which is an important element of many video compression algorithms. The second application is SUSAN [20], which is an edge detection and filtering application used in for example medical applications. The third application is an MP3 audio decoder.

Table 1. Configuration of SimpleScalar

Parameter	Value	Parameter	Value
Issue width	4 insts/cycle	Functional units	4 Int ALU, 1 Int Mult/Div 4 FP ALU, 1 FP Mult/Div
Inst. fetch queue size	4 (in insts)		
Branch predictor	Bimodal 2048 Table	L1 I/D cache	16 KB, 4-way, 32 Byte blocks, 1 cycle
RUU/LSQ size	16/8	Memory latency	18 cycles

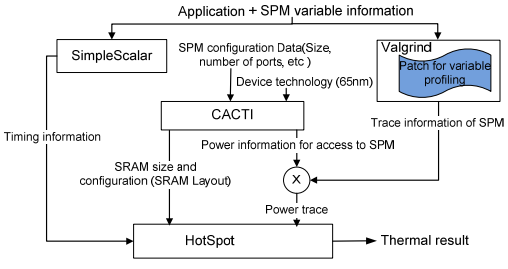

Fig. 5. Tool flow for thermal simulation

A. *Motion Estimation*

The motion estimation application was compiled using the PISA compiler and then simulated with the sim-outorder simulator of SimpleScalar. Using these tools, we determined that the processing time of a single macro-block is 1.4 ms. Therefore, we set the duration of each simulation cycle in HotSpot to 1.4ms. In the motion estimation application, each image window is composed of 4x4 macro blocks. Therefore, we set the size of the SPM to the size of the window, which is equal to 4KB. For simplicity, we assume that the size of one memory bank is equal to one macro block. Since this application exhibits regular access patterns, we use it to evaluate our address mirroring technique. Figures 6 and 7 show the average temperature within each of the memory banks of the SPM when running the motion estimation application. For readability, the line corresponding to one memory bank, the bank storing macro block 4, is changed to a bold, dashed line labeled MB4. Fig. 6 shows the results when the SPM is surrounded by air and Fig. 7 shows the results when the SPM is surrounded by inactive silicon (SPM placed in the center of the die). The results of these two figures determine the efficiency range of the proposed mirroring technique. Fig. 6.a shows the result of a baseline solution where no thermal aware address mirroring is applied and Figure 6.b shows the result when using the proposed method. The horizontal axis in these figures represents a time step within the application execution (each step is equal to the processing of one macro block). We simulated the application for 32 steps in which each memory bank is accessed two times (first access occurs in between time step 1 and time step 16 and the second access occurs between time steps 17 and 32). We evaluated the system based on the second access to each memory bank, because we see from the graph that the transient effects have disappeared. In the first access to the memory banks, the temperature of a memory bank traverses from its transient situation to a steady state. This steady state behavior occurs from the second access to the bank onwards. Fig. 6.a shows that when a memory bank is accessed, its temperature will increase. Once the accesses have been completed, it starts to cool down. For example when the processing of macro block 4 is started, its temperature increases and when processing other macro blocks its temperature decreases. By using the address mirroring technique, we distribute the accesses to one memory bank across the whole SPM. As a result, the whole SPM will experience even accesses over time. Therefore, it shows an even temperature when running the application. Another outcome that can be observed from Fig. 6 is that macro blocks which are placed at the corner of the SPM (in Fig. 6.a macro block 1, 4, 13, and 16) experience a higher temperature in comparison to the macro blocks that are not placed at a corner of the SPM. This is related to the different thermal dissipation flow in the outside banks as compared to the internal banks of the SPM. Internal banks can dissipate temperature through neighboring memory banks, but outside banks must dissipate the heat through an insulator (the surrounding air). By distributing accesses using our technique, this effect is reduced and an even distribution of temperature can be achieved. In Fig. 7, a similar experiment is repeated with an SPM surrounded by inactive silicon. As is clear from Fig. 7.a, all memory banks show similar temperature trends, because outside memory banks are also in the neighborhood of inactive silicon, like internal memory banks. When comparing the results of Fig. 6 and 7, it is clear that the second situation has a lower average temperature due to the better heat transfer in silicon in comparison to air. The peak temperature is reduced up to 10.0°C and at least 3.9°C in the motion estimation application and its thermal cycling (i.e. a repeating difference between maximum and minimum temperature) is reduced from 14.8°C to 0.1°C in the best case and from 5.8°C to 0.1°C in the worst case.

B. *SUSAN*

As a second experiment to evaluate the address mirroring technique, we used the SUSAN application, which is an application from the image processing domain. This application processes consecutively accessed image blocks. These image blocks are placed in the SPM. In this experiment, we allocate 4 KB SPM to this application. The processing time needed for one image block is equal to 0.4 ms. Therefore, this value was set as the simulation cycle time in HotSpot. Fig. 8 shows the average temperature of each memory bank of the SPM when running this application. When comparing the result of the motion estimation application with the results of the SUSAN application (note the different vertical scales), it is clear that our proposed technique can get more gain in applications in which the memory region of the SPM that is accessed by the application, changes with a lower frequency (i.e. if the processing time is larger). The execution time of one data block in SUSAN is less than the execution time for one data block in motion estimation (i.e. 0.4 ms versus 1.2 ms). Therefore, SUSAN inherently already exhibits a better distribution. The peak temperature is reduced up-to 2.6°C in the SUSAN application and its thermal cycling is reduced from 3.8°C to 0.1°C.

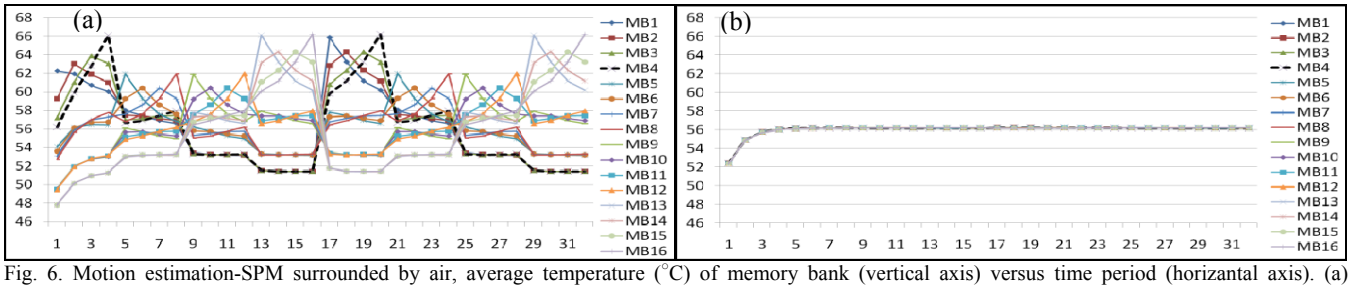

Fig. 6. Motion estimation-SPM surrounded by air, average temperature (°C) of memory bank (vertical axis) versus time period (horizantal axis). (a) temperature result for baseline solution (b) temperature result for address mirroring technique.

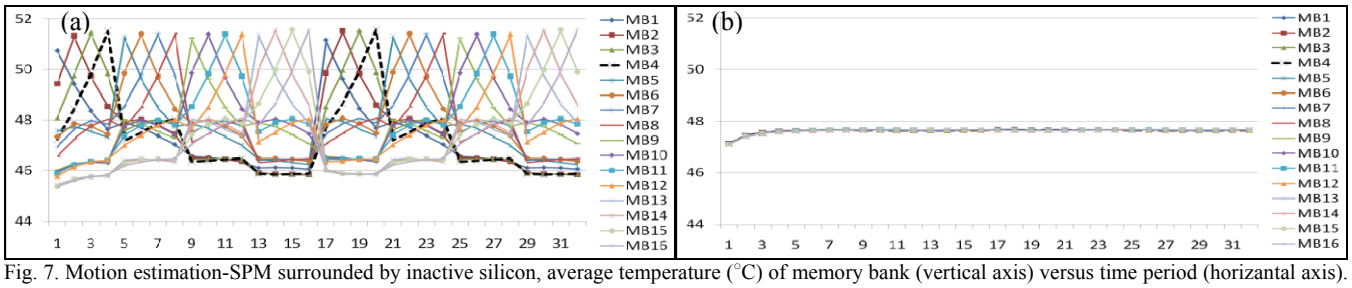

Fig. 7. Motion estimation-SPM surrounded by inactive silicon, average temperature (°C) of memory bank (vertical axis) versus time period (horizontal axis). (a) temperature result for baseline solution (b) temperature result for address mirroring technique.

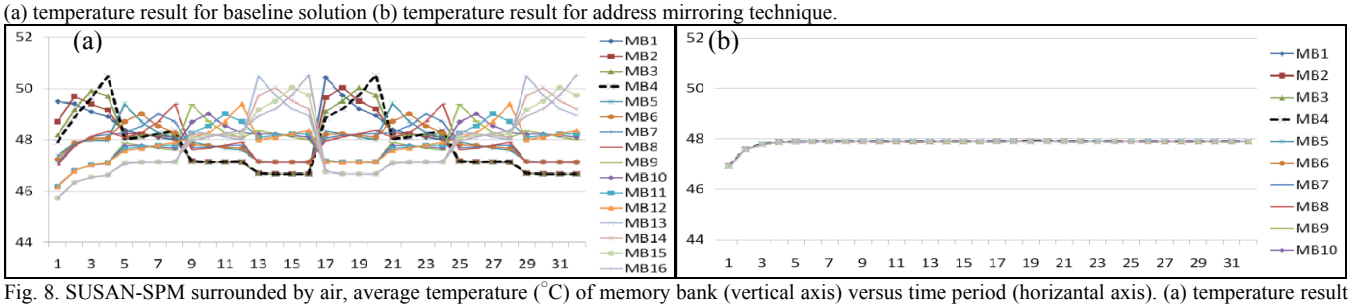

Fig. 8. SUSAN-SPM surrounded by air, average temperature (°C) of memory bank (vertical axis) versus time period (horizontal axis). (a) temperature result for baseline solution (b) temperature result for address mirroring technique.

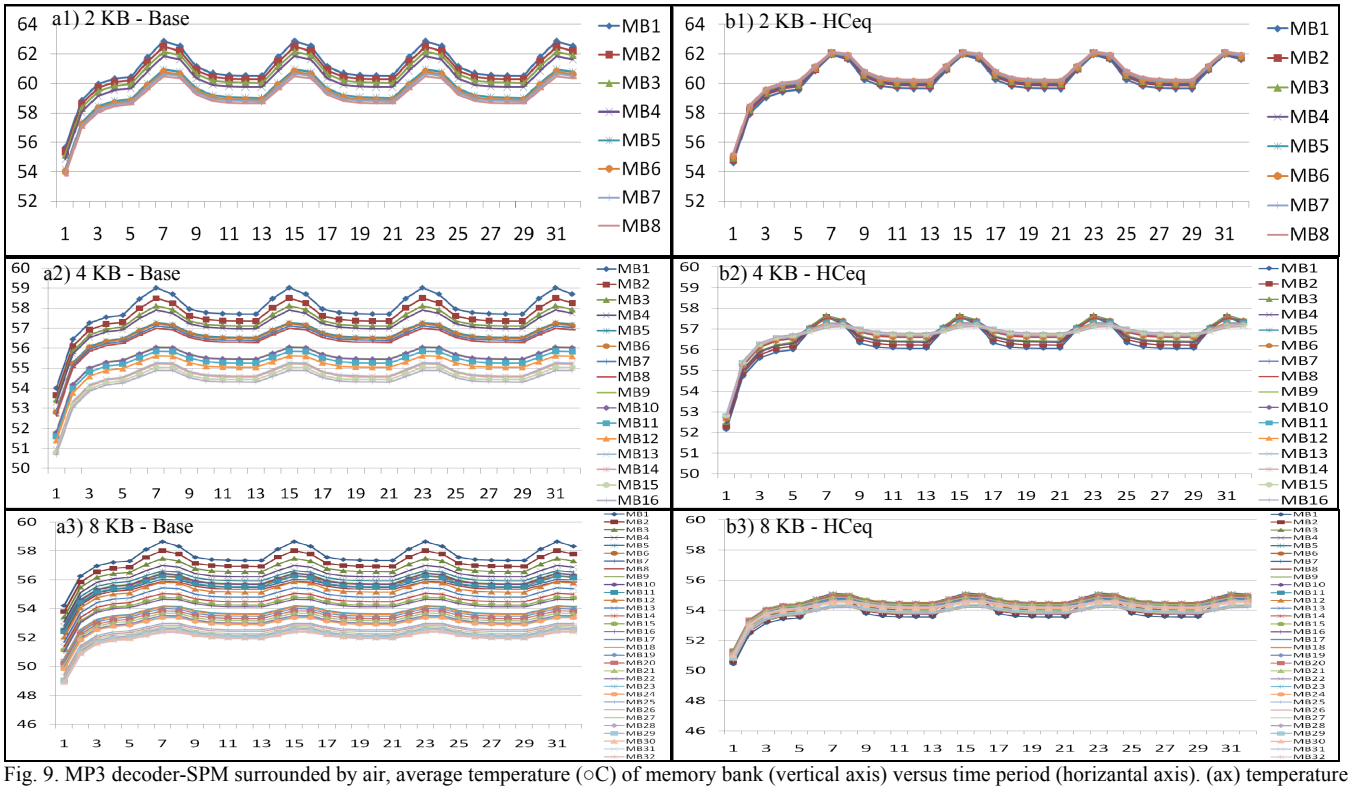

Fig. 9. MP3 decoder-SPM surrounded by air, average temperature (○C) of memory bank (vertical axis) versus time period (horizontal axis). (ax) temperature result for baseline solution (bx) temperature result for HCeq technique.

C. *MP3 Decoder*

As a third application, we studied an MP3 decoder. Compared to the previous two applications, where most of the memory accesses are into a single frame, in the MP3 decoder, several variables are put in the SPM that do not have a regular access pattern. We therefore use it to evaluate our software-based thermal aware address generation. We used the static allocation method from [12] to choose the variables for a 2 KB, 4 KB, and 8 KB SPM. Fig. 9 shows the result of a thermal simulation of this application. This figure shows the temperature results for decoding a sequence of eight audio frames four times. We repeatedly decoded the audio frames to evaluate the temperature in the steady state. As it is clear from Fig. 9, the temperature of memory banks move from their initial state into a steady state after the first repetition (i.e. 1st till 8th's time period in horizontal axis). Fig. 9.a*x* (*x*=1,2,3) shows the results of the thermal simulations for the base line solution. Each memory bank has a different temperature compared to the other banks and the relative temperature does not change (e.g. the temperature of bank 1 is most of the times higher than the temperature of bank 6). These effects are related to the irregular access patterns inside the application. Variables with a higher number of accesses will increase the temperature of a memory bank compared to banks that contain variables with a lower

number of accesses.

Fig. 9.b*x* shows the results of thermal simulations of the SPM when our thermal-aware address generation technique is used to allocate the variables to the SPM. The figure shows that our technique achieves an even average temperature across the whole SPM. Furthermore, our technique is able to reduce the peak temperature of the SPM, up-to 3.5°C for the MP3 decoder. For larger SPM sizes, the gain that can be achieved by HCeq is larger. More less-frequently accessed variables can then be placed in the SPM which means that the ratio of cold variables to hot variables will increase. This will lead to a better variable distribution that the proposed HCeq address generation can exploit.

When applying the address mirroring technique to this application, a result similar to the result for the HCeq technique is achieved. However, address mirroring requires a change in the hardware and cannot be applied in a platform in which the hardware is fixed. The software-based HCeq address generation technique can be applied also to a fixed hardware platform. When the memory accesses to the SPM are regular, it is inevitable to use address mirroring because then the HCeq technique does not work. The HCeq technique is effective when number of access to variables is different. Hence applying this technique to applications with regular memory access pattern has no gain. We also ran simulations combining the two techniques, which confirm the expected result that the two techniques do not interfere. The results are almost identical to the results reported in Fig. 9.

VII. CONCLUSION

SPMs are often used in embedded systems to improve their predictability along with a better performance and a lower energy usage. In streaming applications, it is common to allocate the SPM for streams of data (e.g. image or audio frames). Most streaming applications have a regular access pattern to the memory. In this paper, we propose a technique to reduce the peak temperature and temperature cycling in SPMs when an application with a regular access pattern is executed. Via an address mirroring method, we distribute the memory accesses across the whole SPM evenly. To improve thermal behavior for applications with an irregular memory access pattern, a thermal-aware address generation technique is proposed. It allocates variables to the SPM in such a way that an even power consumption density through the SPM is achieved. A tool flow was developed to measure the application's temperature at the architectural level. In our tool flow, we combine different tools: SimpleScalar, Valgrind, CACTI, and HotSpot. Experimental results show that our techniques are able to reduce the peak temperature in a motion estimation application and SUSAN by respectively 10.0°C and 2.6°C. Thermal cycling is reduced from 14.8°C to 0.1°C in motion estimation and from 3.8°C to 0.1°C in SUSAN by using the proposed address mirroring technique. We applied our thermal-aware address generation technique to an MP3 decoder. This application has an irregular accesses pattern on the SPM. Our technique is able to achieve a 3.5°C reduction in the peak temperature. The two techniques proposed in this paper work well together, improving the SPM thermal behavior for all applications, without incurring silicon overhead or performance loss.

REFERENCES

[1] P. R. Panda et. al. Efficient utilization of scratch-pad memory in embedded processor applications. EDA&T, pp.7-11, 1997.

[2] R. Banakar et. al. Scratchpad memory: A design alternative for cache on-chip memory in embedded systems. CODES, pp. 73-78, 2002.

[3] J.W. Sias et. al. Enhancing loop buffering of media and telecommunications applications using low-overhead predication. MICRO, pp. 262–273, 2001.

[4] W. Liao et. al. Microarchitecture level power and thermal simulation considering temperature dependent leakage model. ISLPED, pp. 211-216, 2003.

[5] W. Huang et. al. HotSpot: a compact thermal modeling methodology for early-stage VLSI design. IEEE Transactions on VLSI systems, pp. 501-513, 2006.

[6] J. Srinivasan et. al. The case for lifetime reliability-aware microprocessors. ISCA, pp. 276-287, 2004.

[7] A. Ajami et. al. Modeling and analysis of nonuniform substrate temperature effects on global interconnects. IEEE Transaction on CAD, pp. 849-861, 2005.

[8] J. C. Ku et. al. Thermal management of on-chip caches through power density minimization. IEEE Transaction on VLSI systems, pp. 592-604, 2007.

[9] K. Patel et. al. Active bank switching for temperature control of the register file in a microprocessor. GLSVLSI, pp. 231-233, 2007.

[10] JEDEC. Failure mechanisms and models for semiconductor devices. JEDEC Solid State Technology Association, 2003.

[11] S. Steinke et. al. Assigning program and data objects to scratchpad for energy reduction. DATE, pp. 409-415, 2002.

[12] F. Angiolini et. al. A post-compiler approach to scratchpad mapping of code. CASES, pp. 259-267, 2004.

[13] M. Verma and P. Marwedel. Overlay techniques for scratchpad memories in low power embedded processors. IEEE Transaction on VLSI Systems, pp. 802-815, 2006.

[14] M. Wang et. al. Improving the reliability of embedded systems with cache and SPM. TSP, pp. 825-830, 2009.

[15] F. Li et. al. Improving scratch-pad memory reliability through compiler-guided data block duplication. ICCAD, pp. 1002-1005, 2005.

[16] P.H.S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. ICCV Workshop on Vision Algorithms. pp.278-294,1999.

[17] D. Burger et. al. Memory hierarchy extensions to simplescalar 3.0. Technical Report TR99-25, Department of Computer Sciences, The University of Texas at Austin, 2000.

[18] N. Nethercote and J. Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. PLDI, 2007.

[19] N. Muralimanohar et. al. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. MICRO, pp. 3-14, 2007.

[20] S.M. Smith and J.M. Brady. SUSAN - a new approach to low level image processing. Journal of Computer Vision, pp. 45-78, 1997.

[21] J.H.L. Pang et. al. Thermal cycling analysis of flip-chip solder joint reliability. IEEE Transactions on components and packaging technologies. pp. 705 – 712, 2001.