Thermal Aware Task Scheduling for 3D ICs

Ara Gevorgyan, Aram Baghdasaryan, Davit Avagyan, Tigran Khazhakyan, and Miljana Milić

***Abstract*—The growing number of elements in integrated circuits increases the need to research and develop new design methodologies. One of the promising technologies is the 3D integrated circuits technique. In this paper, a thermal aware task scheduling methods for processors in 3D integrated circuits are discussed. Several task scheduling algorithms are discussed as well as some possible ways to improve thermal aware scheduling mechanisms. For thermal aware task scheduling, a master-slave structured algorithm is presented.**

***Index Terms*—3D IC, scheduler, Master-Slave, stack**

I. Introduction

The relatively new 3D IC technology has promising future [1]. It allows vertical stacking of several layers of cores and has several advantages in compare with the traditional 2D IC technology. Due to the vertical allocation of dies, interconnects' length could be much shorter then it would be in the traditional one layer allocation. By minimizing the wire length, it is possible to achieve signal edge sharpness and to minimize the signal delays.

Traditionally there are three main options to solve issues and minimize thermal dissipation problem. The first one is based on thermal aware floorplanning. It assumes that during floorplanning stage of design in 3D integrated circuit, the thermal activity of blocks is counted and according to it, the thermal aware floorplanning is performed. Blocks in this case are placed so that the hot blocks are located far from each other and, which in general, keeps the temperature of chip relatively the same in the entire chip. Next approach is the thermal aware placement. This operation is done during the placement stage of circuit elements. The elements are placed based on their switching activity. If an element is very active it should be placed far from another active element. The third method is based on an insertion of special thermal vias. Thermal vias perform thermal energy removal from deep layers of the chip to the upper layers. It is shown that thermal via insertion is the most effective way to remove thermal energy. However, the disadvantage of this method is that thermal vias are usually placed by arrays and occupy much of the useful area of the chip. Another disadvantage of thermal vias are parasitic effects that can happen; for example parasitic capacitance and signal glitches.

All these three methods are hardware based methods. They all assume changes in the physical design of the 3D chip.

In this paper, software approach to minimize thermal issues in 3D integrated circuits will be discussed. This will be done by thermal aware scheduling tasks for multiple processors. In this way, modification of the current chip design can be avoided, while focusing on operations performed by this chip.

II. Prior Work

There are many works investigating thermal challenges and their potential solutions in 3D IC. Black et al. [2] investigated the thermal challenges for vertical stacking of SRAM and DRAM on a processor. It was also suggested to implement processor in two layers and, in that way to separate active and passive logics to different layers. Another approach is to have active and passive logics stacked one after another. Processors are located in the active layer and the passive layer consists of memory blocks like cache and SRAM. Such a design intends to isolate thermally active layers by inserting passive memory layers between them and thus implementing the thermal energy buffer. Several possible arrangements for such mix-stacked structures was suggested (Fig 1) [3].

This illustration shows that there are always active layers in the same vertical stack for 8-core system. Like in 2D designs, chips in 3D designs have better heat conductivity in vertical direction. This assumes that hot layers have more thermal impact on nearby vertical layers than on horizontal ones. The classical thermal model for multilayered systems is show in (Fig 2).

In this model, the active layers are described by their temperature (T) and power (P). Temperatures of the first and the second dies: T1 and T2 respectively can be calculated by the equations (1-3):

$$T_1 = R_{out1}(P_1 + P_2) \quad (1)$$

$$T_2 = R_{out1}(P_1 + P_2) + R_{21}P_2 \quad (2)$$

$$T_n = T_{n-1} + R_{n,n-1}P_n \quad (3)$$

where T indicates the temperature of the core while P stands for the power of the task.

The thermal correlations of two adjusted layers are discussed in [4]. It shows the impact of the hot core to all other cores located in the same stack. The die layers that are located further from the heat sink are usually hotter.

Gevorgyan Ara is with the Synopsys Armenia CJSC, 41 Arshakunyats Avenue ViaSphere Technopark, 0026 Yerevan, Armenia (e-mail: aragev@synopsys.com).

Baghdasaryan Aram is with the Synopsys Armenia CJSC, 41 Arshakunyats Avenue ViaSphere Technopark, 0026 Yerevan, Armenia (e-mail: aramb@synopsys.com).

Avagyan Davit is with the Synopsys Armenia CJSC, 41 Arshakunyats Avenue ViaSphere Technopark, 0026 Yerevan, Armenia (e-mail: davita@synopsys.com).

Khazhakyan Tigran is with the Synopsys Armenia CJSC, 41 Arshakunyats Avenue ViaSphere Technopark, 0026 Yerevan, Armenia (e-mail: khtigran@synopsys.com).

Miljana Milić – University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: miljana.milic@elfak.ni.ac.rs).

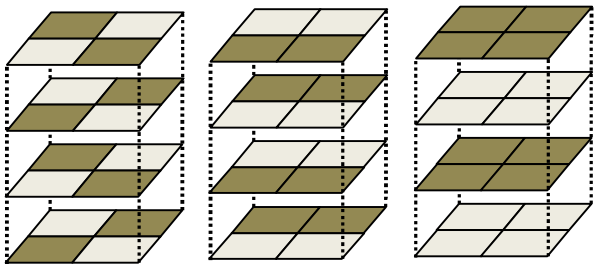

Fig. 1. 3D floorplan options.

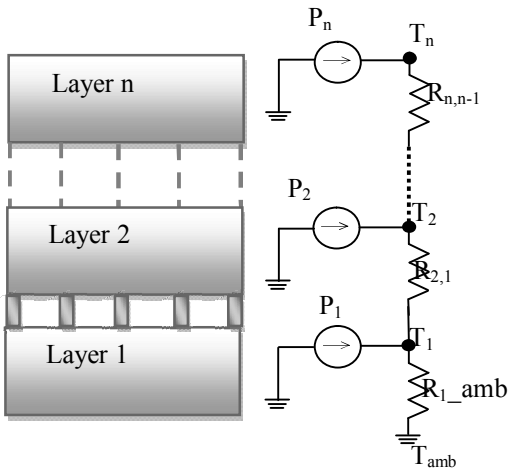

Fig. 2. 3D thermal model.

III. Scheduling Algorithms

In this section, several task scheduling algorithms will be discussed, along with their advantages and disadvantages from thermal awareness point of view.

In order to cool down the hot cores, a dynamic thermal management (DTM) technique [5] is introduced. This mechanism has a very fast response. When core temperature rises above threshold, DTM core reduces power. The application of this method can worsen the performances of the circuit.

In the baseline scheduler, each core has its task queue, which contains running tasks for this core. This scheduler uses two lists; active and expired. Active list contains all running tasks, and expired list contains tasks whose execution time is expired. According to some criteria, core selects the running task from this queue. When the execution time expires, the task is moved to the expired list. When all tasks are placed in the expired list, the scheduler exchanges tasks between the expired list and the active list. The time duration while each task is in the active list, depends on the task priority. Each task is in the active list about 10-200ms of CPU cycle quota. By default, the core switches tasks every 100ms. This algorithm is very simple but it has a risk of putting two hot tasks in the same core stack. Large delay between switching tasks worsens the thermal condition in the chip.

Another schedule algorithm is Round-Robin (RR) [6]. Scheduler adds a new task at the end of the queue that contains all tasks. When task execution time is expired scheduler put this task at the end of queue. The scheduler always selects task from the beginning of the queue. After N iterations (N is the number of cores), each task has been executed in the core for one interval. This mechanism helps draining the high temperature from the cores. All other methods are trying to balance temperature among all the cores. Due to this, temperature in the core stacks rises or decreased very rapidly, which cause thermal emergencies. For avoiding thermal emergencies, the scheduler should balance among core stacks [7].

Another mechanism to cool down the cores is Temperature Balancing by Core [8]. This method arranges tasks depending on the cores' temperatures and tasks consuming power. The Scheduler sorts all tasks by their consuming power and cores by their temperatures. Then the scheduler executes the lowest power-consuming task at the hottest core. This algorithm repeats until all tasks are accomplished. Such a way for distributing temperature is better than the RRs. If one core in the stack is hot, then the neighboring cores worm-up due to the vertical temperature distribution between cores. In that way, temperature differences between cores are kept small. The scheduler will execute less power consuming tasks on the hotter cores. After this, hotter core stacks will have larger temperature drop, while colder core stacks will have a temperature rise that can cause thermal emergencies. In this case, better solution for reducing the temperature is the RR algorithm.

IV. Thermal Aware Scheduling

The scheduling algorithms that were discussed in the previous section perform task scheduling based on task priorities, position in active tasks queue, priority queue etc. Neither of these algorithms takes into account power consumption of tasks. This can lead to heat dissipation problem in the stack and though the entire chip can become useless. The proposed algorithm that will be discussed in this section is the modified version of the priority queues based algorithm, but it considers the values of the task's power consumption also.

Cores that are located in the same stack have strong thermal correlation and if one of the cores becomes too hot it can have effect on the nearby cores too. In that way they will become hot too and the scheduling will not be optimal for them. In addition, it should be taken into consideration that layers that are located near the heat sink are usually colder than the ones that are located far from it.

Considering these two factors, a task scheduling algorithm based on their power activity was developed. The power activity of the task is done based on the performance counters

probing in processor [8]. It is assumed that each core is equipped with such counters. Since the temperature of the layer is dependant on the layer's location regarding the heat sink, though the layers have special relative hotness values (RH), which express their position depending heat. These argument's values depend, not only on the layers location, but also on nearby non-active layers, like SRAM or cache memories.

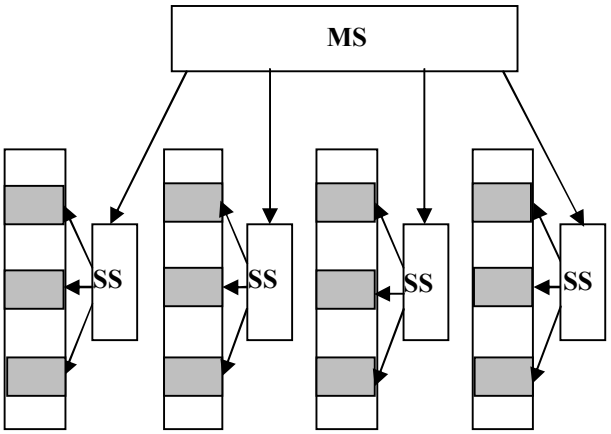

Fig. 3. Master-Slave scheduler structure.

The scheduler has master-slave structure (Fig. 3). For each stack, there is its own scheduler that performs direct task assignment for the processor. This is a slave scheduler that is driven by the master scheduler. The slave scheduler (Fig. 4) performs task assignment to processors, taking into account their actual temperature, task power consumption and processors RH factor. The following algorithm does the task selection and assignment procedure. At the first step, all processors are placed in the thermal priority list based on the RH factor of each one. After that, the tasks are placed in the active tasks queue and are sorted by their power consumption value in descending order, so that the most powerful task will be placed in the first place. After the sorting procedure, the task assignment begins. The first element of active tasks list is assigned to the core with lowest value of RH. This will allow having the most active task scheduled on the core that has better cooling. The second task should be assigned to the second core in cores list and so on. This procedure will schedule all pending tasks on processors so that the colder the processor is, the more powerful task is assigned to it. When time quota for the task is expired then it is moved to the expired task's list and again placed there in a sorted order. This is done to ensure that task will be moved to active task list next time in log(N) time.

The master scheduler (Fig. 5) is responsible for monitoring the current temperatures of cores and performing according resorting of cores in each task pool. In addition, master scheduler assigns tasks to slave schedulers according to the current state of the whole stack of processors. As it was mentioned earlier, if a stack for one of the processors become too hot, it has thermal correlation with nearby layers and due to this, the nearby cores also can become hot. This can lead to the situation when two hottest tasks are scheduled in the same stack on nearby layers and as a result, the whole stack can become too hot. To avoid such situation the master scheduler analyzes the current temperatures of cores. If there is a core in one of the stacks that is too hot, then even if there is a cold core that can take a hot task, such scheduling is not allowed and the task should be assigned to another core that is located in another stack. The general structure of the master scheduler is shown in Fig. 5. It consists of two main blocks, the first part is performing re-sorting of processors pool and the second part is responsible for thermal aware task assignment of core's stacks.

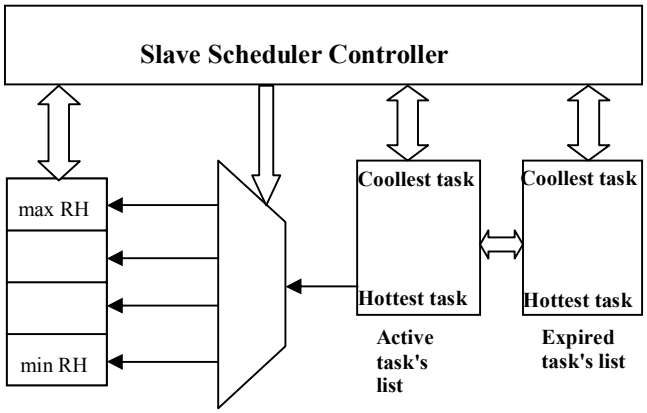

Fig. 4. Slave scheduler structure.

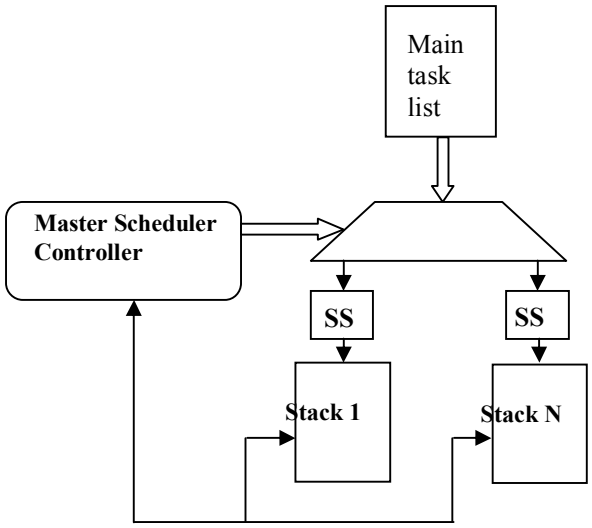

Fig. 5. Master-Slave scheduler structure.

The slave scheduler performs task switching according to the time quota or when the processor's temperature excides threshold value. In this case, the current task's execution is paused and it is moved to the expired queue. This task can

later be rescheduled, on the same or some other processor. If the temperature exceeds the threshold, software sends an interrupt to the master scheduler controller, notifying it about a hot processor stack, and sets a flag for that processor indicating that it was paused due to temperature threshold. No new tasks could be assigned to that core while this flag is set. The master scheduler itself, gets the message sent by the slave scheduler, analyzes the current state of that stack and performs task assignment based on those results. If there is a hot layer it means that nearby layers can become hot very soon, what further indicates that the master scheduler should rearrange the tasks sequence to avoid overheating of the stack. When a new sequence of tasks is calculated, the master scheduler assigns tasks to the slave scheduler and unsets the temperature flag for hot processor. After that, the slave scheduler takes the coldest task and assigns it to the hot processor. This procedure is referred to as a Dynamic Task Assignment.

There is a possible situation when the tasks cannot be rescheduled and there is no task that is cold enough to be scheduled on the hot processor. In such situations the master scheduler keeps the current value of flag until the appropriate task arrives to the main tasks list. While this is happening, it is possible that the hot processor cools down and the temperature value stabilizes in an acceptable range. In this case, the slave scheduler gets an interrupt about that event, analyzes it, unsets the flag and sends notification to master schedulers. After getting this notification, the master scheduler includes those processors in its active processor's list and the new tasks can be calculated and scheduled for that processor.

V. Experimental Results

As an experimental platform a Linux based system was chosen. To simulate the schedulers work and the task assignment special, software was developed. It consists of several modules. The first module simulates the 3D IC, particularly the cores, their RH parameters, temperature threshold values and the passive layers between processors. This module provides a setup and feedback interfaces for the next scheduler module. The second module is designed to simulate schedulers' behavior. It is independent from the scheduler behavior and allows using multiple scheduler models from the standard kernel. This module is responsible for the task scheduling based on the model. In our case, this model consists of two parts: master and slave schedulers. The master scheduler has an interface to get a tasks list form the third module. The third module is responsible for the task list generation. It can generate tasks in two modes: entirely random or with predefined values. The tasks are expressed by their power consumption and by their total running time. The discussed materials and algorithms were implemented using C/C++ programming languages. As a hardware platform, the Intel I5 Core 2.4GHz CPU was used with 4GB of memory. The chosen operating system was Ubuntu OS. To ensure the portability with other Linux systems no third party libraries were used. The application itself is a command line application and could be run from the bash environment. The descriptions of processors and tasks are located in files and are forwarded to the application as command line arguments.

As could be seen from Fig. 6, several comparisons were performed. Different schedulers model were tested. The Master-Slave algorithm shows performance of the same rank compared to the others. The algorithms for task scheduling of the slave scheduler, work in log(N). Together with the master scheduler the total running time for this algorithm is K*Nlog(N), where K is the number of stacks. Such running time result is optimal for this class of algorithms.

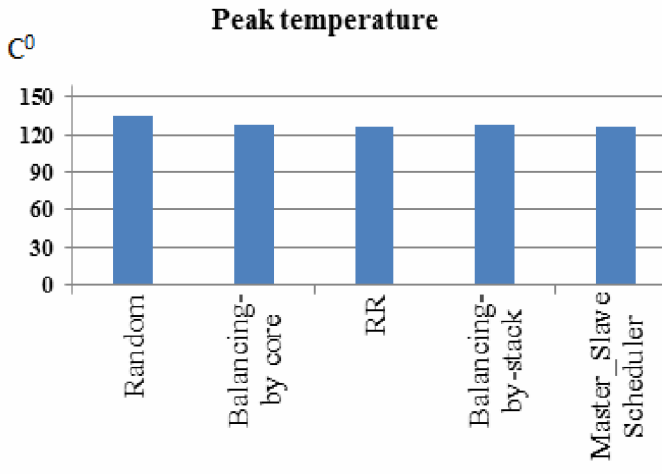

Fig. 6. Experimental results.

VI. Conclusion

In this paper, the problem of thermal dissipation in 3D integrated circuits was discussed. To solve this problem, possible hardware and software approaches were suggested. Since the hardware solutions have influence on the physical design, for keeping temperature of 3D IC in allowable boundaries, a software solution was suggested. The proposed task scheduler algorithms have a master-slave structure. The advantages of this algorithm compared to the others are that it allows having two separate schedulers. Slave schedulers perform local scheduling depending on the current temperature of the core and it's RH factor. The master scheduler performs global scheduling based on the information from the whole stack of cores. This solution allows taking into consideration the effect that, when one of the cores becomes hot, the thermal energy can quickly be forwarded to the other cores making them hot too. The next advantage of such an approach is that it has a modular structure and it allows, in case of a need, to change the structure of schedulers with minimal impact on the other scheduler and the entire design. The disadvantage of such a method is that it uses more resources compared to the other algorithms and may be slower for real time systems.

REFERENCES

[1] W. Topol, D. C. La, Tulipe Jr., L. Shi, D.J. Frank, K. Bernstein, S.E. Steen, A. Kumar, G.U. Singco, A.M. Young, K.W. Guarini, M. Leong, "Three-dimensional integrated circuits", IBM J. Research and Development, vol. 50, no. 4/5, pp. 491-506, 2006.

[2] E. Kursun, C.-Y. Cher, A. Buyuktosunoglu, P. Bose, "Investigating the effects of task scheduling on thermal behavior", the 3rd Workshop on Temperature-Aware Computer Systems, Held in conjunction with ISCA-33, 2006.

[3] M. Awasthi, R. Balasubramonian, "Exploring the Design Space for 3D Clustered Architectures", 3rd IBM Watson Conference on Interaction between Architecture, Circuits, and Compilers (P=ac2), Yorktown Heights, October 2006.

[4] C. Sun, L. Shang, R.P. Dick, "Three-dimensional multiprocessor system-on-chip thermal optimization," Proc. International Conference Hardware/Software Codesign and System Synthesis, pp. 117-122, 2007.

[5] J. Yang, X Zhou, M. Chrobak, Y. Zhang, L. Jin, "Dynamic Thermal Management through Task Scheduling," IEEE International Symposium on Performance Analysis of Systems and software , ISPASS'08, pp.191-201, 2008.

[6] P. Brucker, *Scheduling Algorithms*, Fifth Edition, Springer Press, 2007.

[7] C. Zhu, Z. Gu, L. Shang, R.P. Dick, R. Joseph, "Three-dimensional chip-multiprocessor run-time thermal management", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, V.27, no.8, pp. 1479-1492, 2008

[8] J. Choi, "Thermal-aware Task Scheduling at the System Software Level, " ISLPED'07, August 27-29, Portland, pp. 213-218, 2007.