Dynamic Thermal Management in 3D Multi-Core Architecture through Run-time Adaptation

Fazal Hameed, Mohammad Abdullah Al Faruque and Jörg Henkel
Karlsruhe Institute of Technology, Chair for Embedded Systems, Karlsruhe, Germany
{hameed, mohammad.faruque, henkel} @kit.edu

ABSTRACT

3D multi-core architectures are seen to provide increased transistor density, reduced power consumption, and improved performance through wire length reduction. However, 3D suffers from increased power density, which exacerbates thermal hotspots. In this paper, we present a novel 3D multi-core architecture that reduces processor activity on the die distant to the heat sink and a core-level *dynamic thermal management* technique based on the architectural adaptation, e.g. dynamically adapting core-resources depending on diverse application requirements and thermal behavior. The proposed thermal management technique synergistically combines the benefits of the architectural adaptation supported by our 3D multi-core architecture with *dynamic voltage and frequency scaling*. Our proposed technique provides 19.4% (maximum 24.4%, minimum 15.5%) improvement in the instruction throughput compared to the state-of-the-art thermal management techniques [4, 5] applied to the *thermal-aware* 3D processor architecture without considering run-time adaptation [10].

1. INTRODUCTION AND RELATED WORK

As compared to the transistor delay, interconnect delay and power do not scale at the same rate with each technology node [1]. Three dimensional (3D) integration provides a potential solution to address the wire scalability by reducing communication penalties (by reducing interconnect length) and power (by reducing capacitance) through stacking. However, 3D suffers from increased power density and faces thermal challenges [9, 10, 21].

Heat generation/power dissipation necessitates expensive cooling systems and is a major problem in the 3D multi-core systems. A significant body of work studied [2, 4, 5] *dynamic thermal management* (*DTM*) for the 2D multi-core architecture to tackle the problems of thermal hotspots. These techniques include *dynamic voltage and frequency scaling (DVFS)*, *task migration, fetch gating, decode throttling, speculation control, stop & go,* and *clock throttling* [4, 5, 6]. These techniques are effective for the 2D *DTM*, but they suffer performance degradation in the 3D due to heterogeneous cooling of the 3D stacked layers. Inter-die thermal correlation is much stronger compared to intra-die thermal correlation in the 3D integration [8, 9]. Thermal management in the 3D multi-core architectures solely based on these techniques does not achieve optimal *DTM* because it does not consider vertical heat spreading which is more significant in the 3D integration technology as compared to the planar technology [8, 9]. A useful combination of these techniques will be much more effective for the 3D multi-core thermal managements, if they are tailored wisely to reduce vertical heat spreading [9].

In the context of 3D processor architecture, previous work has investigated performance benefits, energy savings, and thermal challenges. Earlier studies have proposed *thermal-aware* processor architectures [10] by partitioning the components of processing cores (arithmetic units, register file, instruction scheduler, cache, re-order buffer, etc) across multiple dies for latency and energy reduction. It places a majority of switching activity on the die closest to the heat sink (*Die1* as shown in Figure 1). Research work in leveraging the 3D integration for multi-core architecture focuses on stacking cache and *DRAM* on cores to provide latency benefits and improved memory bandwidth [11, 12]. Research work presented in [9] for the 3D multi-core *DTM* has investigated task migration and *DVFS* using a combination of hardware/software techniques governed by thermal management policies by stacking conventional 2D processor dies on the top of each other.

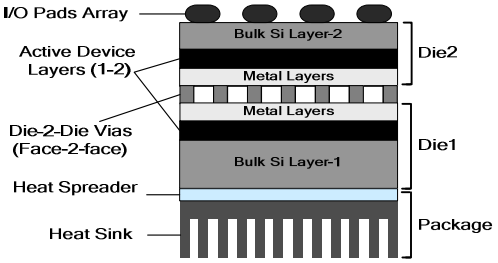


Figure 1: A face-2-face 2-layer 3D stacked die [9]

Heterogeneous cooling efficiencies[1] of the 3D-stacked layers raise the importance of efficient thermal management to maximize the performance of the 3D multi-core architectures under safe thermal limits. The basic tenet of our 3D multi-core *DTM* technique is based on reducing processor activity on the die distant to the heat sink to reduce vertical heat spreading.

Our novel contributions in this paper are as follows:

1. We propose a novel 3D *thermal-aware* processor architecture that utilizes several techniques exploiting integer and floating point program behaviour by partitioning a single core across two dies to reduce dynamic and leakage power on the die distant to the heat sink. Our *thermal-aware* processor architecture focuses on distributing majority of the activities close to the heat sink.
2. We exploit dynamic run-time architectural adaptation (dynamically adapting core-resources depending on diverse application requirements), which reduces the thermal hotspots by further reducing the impact of inter-die thermal correlation. We combine the benefits of fine-grained architectural adaptation[2] (with minimum performance penalty) with the coarse-grained sensor based adaptation[2] that proactively reduces the occurrences of thermal hotspots.
3. We propose efficient 3D *DTM* technique that exhibit high level of thermal safety (keeps the temperature under safe thermal limits) with minimal performance degradation by coupling the state-of-the-art *DVFS* and *migration* with fine/coarse grained adaptation.

[1]Die closer to the heat sink has better cooling efficiency than the die distant to the heat sink [9].

[2]Fine-grained adaptation is triggered by the performance counters [7] and has a lower performance penalty compared to the coarse-grained adaptation. Coarse-grained sensor based adaptation (*SBA*) is triggered by the temperature sensors, which helps to reduce thermal emergencies.

978-3-9810801-7-9/DATE11/©2011 EDAA

2. 3D Processor Architecture

In this section, we describe our novel adaptive *thermal-aware* 3D processor architecture that exploits integer and floating point behaviors by partitioning a single core across two dies to provide simultaneous latency reduction, performance improvement, and vertical heat reduction.

2.1. Thermal-aware 3D Integer Execution Unit

The motivation behind our 3D *thermal-aware* integer execution unit is based on the experimental observation that a small percent of integer instructions require two architectural source operands [15]. These instructions include load instructions, instructions that uses immediate fields, NOP, FTOI (*floating to integer instructions*), and ITOF (*integer to floating*) instructions, etc. Figure 2 shows the percent distribution of the integer instructions that require zero, single, and two source operands from the integer register file for SPEC2000 integer benchmarks.

Conventional integer execution unit for an alpha-21264 processor core contains two clusters (*Cluster0* and *Cluster1*) with two copies of duplicated register file [13]. Each cluster can issue a maximum of two integer instructions in a single clock cycle. The register file of each cluster requires four read ports (to issue two integer instructions in a clock cycle) and six write ports.

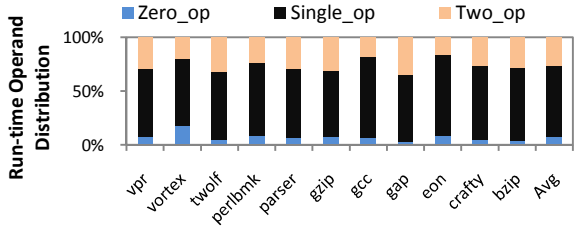


Figure 2: Operand distribution (SPEC2000 Benchmark)

Figure 3 shows our proposed *thermal-aware* integer execution unit. The clusters are vertically stacked on top of each other by deploying a 3D port-split register file [14] shared by the vertical adjacent clusters. The port-split register file not only reduces the footprint of the register file but also provides simultaneous improvement in latency and energy savings. Integer instructions with two source operands are only steered to *Die1* while the integer instructions with zero/single operands can be guided either to *Die1* or *Die2*. The port-split register file on *Die2* contains fewer register file read ports as compared to *Die1*.

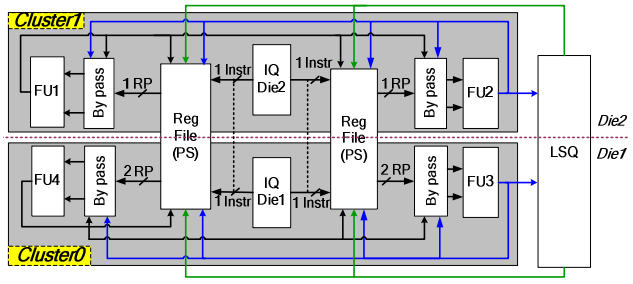


Figure 3: *Thermal-aware* 3D integer execution unit

Figure 4 shows various fields of a single issue queue entry with wake-up logic [15]. We partition the instruction scheduler based on entry partitioned issue queue design in [10, 16] with half of the entries placed on each layer (*Die1* and *Die2*). This approach reduces the length of the tag broadcast bus as compared to a scheduler implementation in the conventional planar technology. Issue queue entries on *Die2* have a reduced payload size and tag comparators as compared to *Die1* in our proposed approach (single operand instructions will require a single comparator for wake-up instead of two). The primary advantage of removing tags from the issue queue entries on *Die2* is the reduction of the load capacitance during instruction wake-up. At the same time, each *issue queue* (IQ) entry size is reduced which reduces the tag bus length to further minimize the power on *Die2*. Figure 4(b) shows various fields of a single issue queue entry wake-up logic for the scheduler design on *Die2* in our proposed approach. Additionally, the arbiter logic implemented on each die is also simplified providing further power benefits. The footprint of the register file is reduced on *Die2* (reduced number of ports as compared to *Die1*) which shortens the length and complexity of the bypass network. Hence, it provides power reduction in the bypass logic (reduces power in the tag and data broadcast bus) compared to previously proposed entry partitioned issue queue design [10].

2.2. Thermal-aware 3D Integer Rename Table

Integer rename table (*IRT*) is implemented as a 3D port-split structure [10,14] by placing the ports corresponding to each instruction on each die. To reduce the power in the *rename table* in our proposed approach, a maximum number of double/single operand instructions that can be steered to *Die1*/*Die2* is limited to two in a single cycle, respectively. The *rename table* on *Die2* contains fewer read ports as compared to *Die1* as only zero/single operand instructions are steered to *Die2* to provide power savings in bitlines, wordlines, and row decoding logic. Previously proposed [10] *rename table* (for a decode width of four) will require 8 read ports and 4 write ports. Our proposed *thermal-aware* 3D *rename table* requires 6 read ports (4 read ports on *Die1* and 2 read ports on *Die2*) and 4 write ports (2 write ports on each layer). The reduction of ports on *Die2* for the *rename table* reduces the instruction payload size (the instruction payload on *Die2* does not require the physical register tag for the second source operand) and hence it reduces the dispatch interconnect size providing a power benefit on *Die2* compared to previously proposed integer rename table [10].

2.3. Thermal- Aware 3D Floating Point Unit and SRAM Structures

Our 3D *thermal-aware floating point* (FP) execution unit is based on the experimental observation that a FP multiply unit is rarely used as compared to a FP add unit (on average, the FP multiply instruction has a 9% distribution for FP SPEC2000 benchmarks based on experimental observation). Our *thermal-aware* 3D FP execution unit places the least utilized multiply unit with one store pipeline on *Die2*, while it places the add unit with the other store pipeline on *Die1*. *Add, sqrt,* and *divide* instruction are steered to *Die1* (non-pipelined divider unit and non-pipelined square root units are associated with the adder pipeline) [13], while the *multiply* instruction is steered to *Die2*. Store instructions can be steered either to *Die1* or *Die2*. To further reduce the power on *Die2,* the FP i*ssue queue* on *Die2* has reduced entries as compared to *Die1* in order to reduce the thermal impact. We implement r*eorder buffer* (*ROB*), instruction cache, data cache, branch predictor, *load/store queue (LSQ)*, *instruction fetch queue (IFQ), instruction translation look-aside buffer* (*ITLB*), and *data TLB* using previously proposed 3D stacking organization [10, 17, 21].

2.4. Fine-grained Architectural Adaptation (RA)

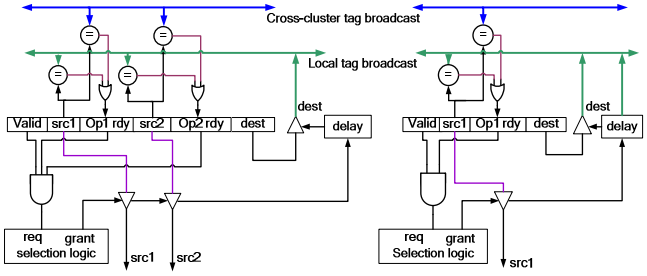
An Integer execution unit is the most critical component with intense power densities [4, 8]. We propose a *thermal-aware* run-time architectural adaptation, which is inspired from previous work [7] to reduce the 3D vertical heat spreading. We define five different states for run-time adaptation.

- IW4 (*Integer issue width*-4) : In this mode, the integer execution unit runs in normal mode with a maximum of two inte-

ger instructions (zero /single /two operand instructions) steered to *Die1* and a maximum of two integer instructions (zero /single operand instruction) dispatched to *Die2* in a clock cycle.

- IW3: In this mode, the *integer issue queue* on *Die1* issues two instructions per cycle, while the *integer issue queue* on *Die2* issues one instruction per cycle and hence not precharging the corresponding register file read and write ports.
- IW3_RP (reduced power): This mode supports all of the *thermal-aware* features of the IW3 state. Additionally, one integer functional unit and half of the issue queue entries on *Die2* are disabled to provide further power saving. We employ a segmented issue queue similar to the design mentioned in [18] that contains two issue queue partitions on *Die2*, each of which may be enabled/disabled at run-time.
- IW2: In this state, the *integer issue queue* on *Die1* issues two instructions per cycle. The *integer issue queue* on *Die2* issues single instruction per cycle until the issue queue becomes empty. The decode width is scaled to two integer instructions, all of which are steered to *Die1*. The i*nteger rename table* ports on *Die2* are not precharged.
- IW2_RP: This state supports all of the *thermal-aware* features of the IW2 state. Additionally, all of the issue queue entries and functional units on *Die2* (IFU_{Die2}) are disabled and do not issue any instruction.

Figure 5 shows a run-time architectural adaptation *finite state machine* (FSM) for adapting core resources at regular interval by monitoring the average integer instructions issued per cycle (I_{IPC} = N_{IPC} / I_{RA}) over a window of I_{RA} cycles. At the beginning of the invocation period, the counter N_{IPC} (number of integer instructions issued over a window of I_{RA} cycles) is reset. N_{IPC} is incremented each cycle by the number of integer instruction issued during that cycle. Run-time adaptation implemented in the processor hardware detects a phase change to adapt the integer execution unit from one *thermal-aware* state to another. IW2 and IW3 states are associated with five bit saturating counters. When the integer execution unit remains in the state IW2/IW3 for 32 consecutive invocation periods then the signal OIW2/OIW3 (Figure 5) is asserted.



(a) (b)

Figure 4: Instruction wake-up logic [15] for (a) single issue queue entry on *Die1* (b) single issue queue entry on *Die2*

3. 3D Dynamic Thermal Management

The primary target of our 3D multi-core *DTM* is to provide an efficient hardware/software mechanism that is continuously engaged to adapt with the change in application and thermal behavior over time. Figure 6 shows an overview of our 3D multi-core DTM technique by combining adaptation with *DVFS* and migration. The architectural adaptation is triggered by a performance counter (I_{IPC}) to dynamically adapt core resources with the change in application behaviour, and has a reduced performance penalty with significant energy reduction when scaled at much finer granularity [7], which is leveraged to reduce the power on *Die2*. Throttling (clock throttling, decode throttling, etc.) and *DVFS* have been employed in the past [4, 5] for thermal management. We exploit the *sensor based adaptation* (*SBA*) that relies on the temperature sensors to reduce the 3D vertical heat spreading. The *SBA* proactively reduces the occurrences of thermal emergencies but cannot eliminate it completely. Our 3D multi-core *DTM* uses *DVFS* as a final shield to keep the temperature under safe thermal limits. *DVFS* guarantees thermal safety through quadratic reduction in power saving with linear degradation in performance. The components scaled by the *SBA* are (1) the *integer issue queue* and (2) *instruction fetch queue*. Based on a *proportional integral* (PI) controller output, the *SBA* will be engaged /disengaged by putting the core into throttle/normal mode (TM/NM). In a *throttle mode*, half entries of the above components will be disabled on *Die2* (half entries are enabled and half entries are disabled on *Die2*). Throttling will put the core in IW3_RP state (section 2.4 and Figure 5). Scaling theses structure not only reduces the dynamic and leakage power dissipation on *Die2* but also reduces the activity factor to control the worst case temperature. The architectural parameters adapted in the TM mode are (1) the *decode width* (*DW*) is scaled to three instructions per cycle (a maximum of two instructions on *Die1* and one instruction on *Die2*) instead of four instructions per cycle; (2) the *integer issue width* (*IW*) is scaled to three integer instructions (maximum of two integer instruction issued on *Die1* and one instruction on *Die2*) (3) fetch gating [5] to stall the instruction fetch unit after every *L* cycles. In the TM, the core can only switch between IW3_RP and IW2_RP states. When the core resumes the NM, it can enter the high power state IW4, when the current state is IW3_RP and I_{IPC} exceeds 2.0.

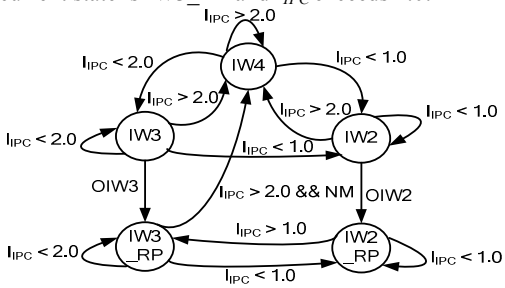


Figure 5: *Thermal-aware* states for the run-time architectural adaptation. For normal mode (NM) see Section 3

Thermal management using a *global migration* (*GM*) helps to balance the temperature by migrating the tasks, and thus change the mapping of threads to cores in case of temperature imbalance. *GM* implemented in software has improved performance benefits, when invoked at coarser granularity [4, 9]. *GM* is invoked at regular interval that finds the core with the maximum temperature imbalance and the suitable thread (on some other core) that would be able to balance the temperature of the core with maximum temperature imbalance. We assume that the cores involved in migration suffer a penalty of 100 µsec similar to the assumption in [4, 9]. The details of *GM* may be found in [4].

Figure 6 shows our 3D *DTM* for a set of *M* cores and *N* functional units. I_{RA}, I_{SBA}, I_{DVFS}, and I_{GM} are the invocation interval for the runtime architectural adaptation, sensor based adaptation, *DVFS*, and *GM* respectively. $T(C_i, U_j)$ and $A(C_i,U_j)$ are the temperature and access statistics of the j^{th} unit for the i^{th} core. Each core gathers runtime processor activity factors through the use of performance counters [4, 9] and temperature through thermal sensors to the *GM* routine. The migration decisions are communicated back to the processor cores by GM routine (shown by arcs) according to the algorithm shown in Figure 6. We provide a 3D *DTM that* synergistically combines the benefits of orthogonal *DTM* schemes involving the core-level run-time architectural adaptation (low performance penalty with energy and temperature reduction),

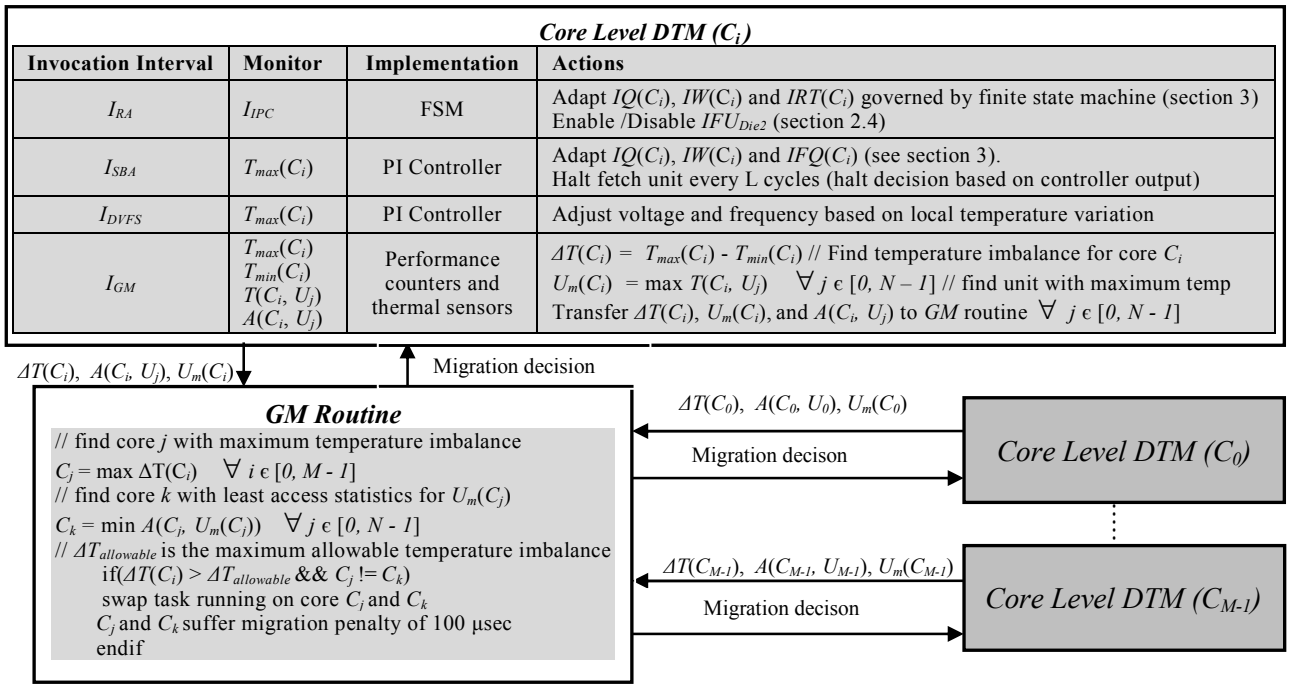


Figure 6: Overview of the 3D *DTM*

core-level *SBA* (reduces thermal emergencies through vertical heat reduction), core-level *DVFS* (keeps temperature under safe thermal limits), and *global-level migration* (provide temperature balancing). Our multi-granularity 3D *DTM* (see Figure 6) employs a large invocation interval (coarse-grained *DTM*) for the *DTM* scheme with a high performance penalty and small invocation interval (fine-grained *DTM*) for the scheme with low performance penalty, i.e. $I_{GM} > I_{DVFS} > I_{RA}$.

The hardware for the run-time adaptation requires a 10-bit counter for N_{IPC} (for I_{RA} = 256 cycles with maximum integer issue width of 4), a 10-bit comparator, two 5-bit counters for the signal OIW2/ OIW3, and a 3 bit counter for saving the current state of the system. Area, dynamic and leakage power of the hardware are incorporated. The most frequently switching 10-bit counter for N_{IPC} with a comparator and FSM logic is placed on *Die1* and the least switching counters for OIW2/OIW3 are placed on *Die2* (only two die-2-die vias needed for the signal OIW2 and OIW3). *SBA* and *DVFS* are implemented using the *proportional integral* (PI) controller with the *SBA* having a high invocation interval ($I_{SBA} > I_{DVFS}$) and a low temperature threshold ($T_{SBA} < T_{DVFS}$). *DVFS* and *SBA* require two PI controllers, which require special registers with *add* and *multiply* instructions inserted into the pipeline to compute the controller output. Since the controllers are invoked infrequently, their contribution to performance and power is neglected in our simulation. Our architecture uses the state-of-the-art *DVFS* hardware supported by modern microprocessors [4]. Our *thermal-aware* instruction steering assigns instruction to each die keeping in view the occupancy of the integer issue queue, the *thermal-aware* states, the SBA mode (NM/TM), and the instruction types (zero/ single/ two operand instruction). The steering logic has a small area (require additional multiplexer and adder) and power overhead.

4. Experimental Set-up

We use a modified version of the simplescalar as performance simulator [20] and the McPat [19] to estimate access latencies, dynamic energy, leakage power, and area of different micro-architectural components. The McPat is augmented to support various 3D structures including the 3D port-split register file and intra sub array partitioning across different dies [10, 14, 21]. Our 3D integrated design assumes a die wafer level face-2-face topology with copper bonding. We assume a die-2-die via size of 1 µm. The length of the vertical interconnect between the two layers is assumed to be 10 µm. The per unit resistance and capacitance of the die-2-die via is determined in the same way as traditional on-die interconnects.

Previous research has analyzed that the integer issue queue and the by-pass logic are the critical components to choose the processor clock frequency [3]. Similar to the assumption in [10, 15], we consider that the issue queue is the cycle time limiter to determine the processor clock frequency. SPEC2000 benchmarks are used to generate power traces and simulation statistics for 500 million instructions after architectural warm-up of 100 million instructions. These traces are generated for each voltage and frequency levels (voltage range 0.7 – 1V with 8 levels) and two adaptation modes (TM/ NM). For each voltage level, we calculate access time for each micro-architectural component to determine its latency. The frequency at each voltage level is calculated based on the *integer issue queue* access time [10, 15]. Latency, dynamic and leakage power of each component are fed to an integrated performance/power simulator (modified simplescalar and McPat) at each voltage level. The simulator generates traces of power (power for *Die1* and *Die2* averaged over interval of 10,000 cycles) and performance statistics (starting committed instruction number, ending committed instruction number, number of instructions executed, access statistics for each component for the global migration decision, etc).

Using Hotspot5.0 [6] for the 3D circuits, each die is modelled as two layers, the active silicon (1 µm thickness) and the bulk silicon with a 32 x 32 grid resolution. Heat sink is placed near to *Die1*. The power consumed in the die-2-die vias is credited equally to the vertical adjacent units and the interconnect power is credited to the units that they connect in proportion to their respective areas. A starting ambient temperature of 45°C is assumed. The pri-

mary advantage of decoupling performance/power and the thermal simulation is to reduce the simulation time for the 3D multi-core *DTM*. We assume a penalty of 10 µsec per voltage transition.

SRAM Structure	Thermal-Herding [10]	Our Architecture
***Int IQ* (each die)**	16	16 (reduced tag)
FP *IQ* entries (20 entries)	(10/10)*	(13/7)*
***Int Reg File* read ports**	4 (2/2)*	3 (2/1)*
***Int Rename Table* read ports**	8 (4/4)*	6 (4/2)*
***Int IQ* access time (32 entries)**	479 psec	432 psec
Operating Frequecy (1.0V)	2.08 GHz	2.30 GHz
***Int Rename Table* (t_{acc}/L)**	721 psec / 2 cyc	657 psec / 2 cyc
ITLB (128 entries) (t_{acc}/L)	794 psec / 2 cyc	794 psec / 2 cyc
DTLB (128 entries) (t_{acc}/L)	835 psec/ 2 cyc	835 psec/ 2 cyc
Load Queue (32 entries) (t_{acc}/L)	668 psec / 2 cyc	668 psec / 2 cyc
FP *add/mult* Latency	4 cycles	4 cycles
FP *divide* Latency	12 cycles	13 cycles
FP *sqrt* Latency	18 cycles	20 cycles
Int *Add/Shift* Latency	1 cycle	1 cycle

Table 1: Comparisons for each 3D processor architecture
t_{acc}/L : acess time in psec/ latency in clock cycle
Access times calculated at 65nm technology (1.0 V)
*** (*Die1*/*Die2*)**

Our 3D multi-core *DTM* simulator makes thermal management decision (to decide voltage/frequency level, *SBA* and migration) at a sampling interval of 100 µsec after feeding the power values to the thermal model from HotSpot. At the end of each sampling interval, the number of committed instructions for each core is calculated (in case of voltage/frequency transition, the number of committed instructions is adjusted considering the transition penalty overhead at corresponding frequency) and multiplied by the corresponding frequency to get BIPS (Billions of instructions per second) for each core, which are summed together to get the total instruction throughput. The thermal simulation is carried out for 5 seconds. When the performance/power trace for a particular benchmark finishes before this duration, the trace is restarted.

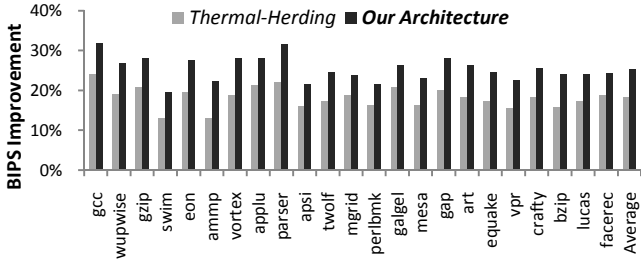


Figure 7: BIPS improvement for the 3D design over the planar design

5. Experimental Results

For the evaluation based on alpha 21264 microarchitecture, we have compared our approach with the state-of-the-art 3D processor architecture namely *thermal-herding* [10], which shares the same philosophy of reducing processor activity on the die away from the heat sink to abate peak temperature and proposed various micro-architectural techniques including a significance partitioned data-path and an address memorization scheme for the load and store queue. The *Thermal-herding* architecture attempts to reduce the dynamic power (by placing frequently switching least significant 32 bit on *Die1* and rarely switching 32 most significant bits on *Die2*) using width prediction. The *Thermal-herding* architecture involves additional overhead of width prediction and suffers from the pipeline stall in case of wrong prediction. Our processor architecture exploits program behavior for the integer instructions to reduce the *integer issue queue* tag comparators and payload sizes, the integer *register file* ports, the *integer rename table* read ports, and the *dispatch interconnect* size on *Die2*, which reduces dynamic and leakage power on *Die2*.

As shown in Table 1, our proposed 3D processor architecture has reduced access time for the *integer issue queue* (through reduced tag comparators and payload size) as compared to *thermal-herding* architecture, which results in the operating frequency increase compared to the *thermal-herding* architecture. Since the *integer rename table* and the *register file* access times are reduced, an increase in operating frequency will not change the integer register file and dispatch latency (dispatch latency is dependent upon integer rename access time and instruction steering logic delay) compared to the *thermal-herding* architecture. FP rename and register file latencies are not effected as they are not the cycle time limiter. FP *sqrt* and FP *divide* latency are increased, but their increase in latency does not have a strong influence on the instruction execution throughput. The primary concern about the integer instruction steering logic is that whether it affects the performance in the form of additional latency due to steering. The integer dispatch latency (defined as the number of cycles required to dispatch an integer instruction to the integer issue queue, which is configurable in our simulator) to a particular clusters depends on the integer rename table access time and the steering logic delay. Since our thermal-aware architecture has reduced number of read ports for the integer rename table (hence delay), therefore, the additional overhead of the instruction steering does not affect the integer dispatch latency compared to the *thermal-herding architecture* [10].

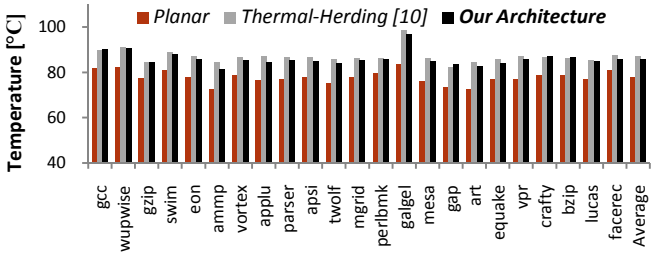


Figure 8: Maximum temperature for the planar and the 3D designs (without *DTM*).

Architectural parameters for the single core simulation are given in Table 1, with L1 instruction and data cache of 64KB and L2 cache of 1.75 MB. Since the FP *multiply* unit (*Die2*) is rarely used compared to the FP *add* unit (*Die1*), we have used a different number of FP *issue queue* entries on each die (13 entries on *Die1* and 7 entries on *Die2*). Figure 7 shows the percent BIPS improvement for the *thermal-herding* [10] and our architecture (with the run-time adaptation without DVFS and migration) over the conventional planar design [13]. On average, our proposed 3D processor architecture increases the performance by 25.3% (minimum 19%, maximum 31% for individual benchmarks), while the *thermal-herding* architecture increases the performance by 18.1% as compared to the planar design. Figure 8 shows the maximum temperature for the conventional planar design [13], the *thermal-herding* [10] and our architecture. With the *thermal-herding*, the worst case temperature increases by 9.2°C and with our architecture, the worst case temperature increases by 8.2°C as compared to the conventional planar design. Our architecture delivers better instruction throughput compared to the *thermal-herding* architecture with reduced worst case temperature. Despite the operating frequency increase, our thermal-aware architecture reduces the worst case temperature through the novel *thermal-aware* integer unit, the FP execution unit design, and the run-time adaptation, as the integer and the FP execution units are

the critical thermal hotspots [4, 8]. Our 3D thermal-aware processor architecture reduces the dynamic and the leakage power on *Die2,* while the *Thermal-herding* architecture attempts to reduce the dynamic power. Leakage power is significant fraction of the total power for the future technology nodes [1]. With the run-time adaptation, we save additional dynamic and leakage power on *Die2* by disabling the *integer issue queue* entries and the functional units with change in application behavior to reduce vertical heat spreading.

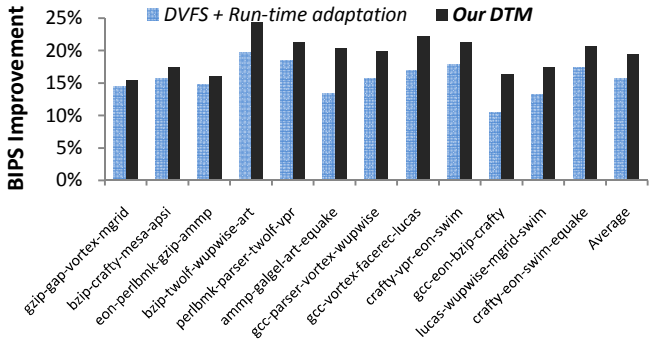


Figure 9: Improvement in the instruction throughput relative to *DVFS* for the *Thermal-Herding* Architecture [10]

Our 3D multi-core *DTM* simulation uses a unified L2 cache of 4MB (M = 4, I_{RA} = 256 cycles, I_{DVFS} = 0.1 msec, I_{SBA} = 1 msec, I_{GM} = 30 msec, T_{DVFS} = 84.2°C, T_{SBA} = 84°C, and L = 3 cycles). T_{SBA} and T_{DVFS} are the temperature threshold for *SBA* and *DVFS* respectively. Figure 9 shows percent improvement in the instruction throughput achieved relative to the state-of-the-art *DVFS* [4, 5] for the *thermal-herding* architecture [10]. Our 3D *DTM* improves instruction throughput by 19.4% (maximum 24.4%, minimum 15.5%) relative to *DVFS* for the *thermal-herding* architecture. Our 3D thermal-aware processor architecture with the run-time adaptation (without *SBA* and *GM*) alone improves the instruction throughput by 15.7% (maximum 19.8%, minimum 10.5%) compared to *DVFS* for the *thermal-herding* architecture. The contribution of leakage power becomes more significant compared to the dynamic power at low voltages [1] (*DVFS* attempts to reduce the power consumption by lowering voltage and frequency).

We have introduced an adaptive thermal-aware 3D processor architecture that dynamically adapts core resources with the change in application and thermal behavior. We synergistically combine the benefits of adaptation, fail-safe *DVFS*, and global migration to achieve better instruction throughput that are substantially higher than what is possible using a single technique. One of the key limitations for the realization of the 3D integration are the die-2-die via sizes constrained by the accuracy of aligning wafer in the bonding process [14]. However, in this work, we assume a face-2-face topology, which provides high via density compared to a face-2-back topology.

6. CONCLUSION

This paper presents a 3D *DTM* technique on top of our novel *thermal-aware* 3D multi-core architecture that synergistically combines the benefits of core-level adaptation (proactively reduces the occurrence of thermal emergencies through vertical heat reduction), *DVFS*, and *global migration*. Our 3D *DTM* technique delivers 19.4% better instruction throughput compared to *DVFS* for existing *thermal-aware* 3D processor architecture. Even the simpler run-time adaptation coupled with *DVFS* outperforms (15.7% better instruction throughput) the *thermal-herding* architecture with *DVFS*. Our proposed *DTM* technique considering various architectural adaptations comes with limited hardware/software overhead.

7. REFERENCES

[1] International Technology Roadmap for Semiconductors, http://www.itrs.net.

[2] P. Chaparro, J. Gonzalez, G. Magklis, Q. Cai and A. Gonzalez, "Understanding the Thermal Implications of Multicore Architectures", IEEE Transaction on Parallel and Distributed Systems, vol. 18, No. 8, pp. 1055-1065, August, 2007.

[3] R. Jayaseelan and T. Mitra, "A Hybrid Local-Global Approach for Multi-core Thermal Management", ICCAD'09: International Conference on Computer Aided Design, pp. 314-320, 2009.

[4] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors", HPCA'01: International Symposium on High-Performance Computer Architecture, pp. 171-182, 2001.

[5] J. Donald and M.Martonosi, "Techniques for Multi-core Thermal Managment: Classification and New Exploration", ISCA'06: International Symposium on Computer Architecture, pp. 78-88, 2006.

[6] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang,S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation", ACM Transactions on Architecture and Code Optimization, vol. 1, No. 1, pp. 94–125, 2004.

[7] R. I. Bahar and S. Manne, "Power and Energy Reduction via Pipeline Balancing", ISCA'01: International Symposium on Computer Architecture, pp. 218-229, 2001.

[8] G. M. Link and N. Vijaykrishnan, "Thermal Trends in Emerging Technologies", ISQED'06: International Symposium on Quality Electronic Design, pp. 625-632, 2006.

[9] C. Zhu, Z. Gu, L. Shang, R.P. Dick and R.Joseph, "Three-Dimensional Chip-Multiprocessor Run-Time Thermal Management", IEEE Transaction on Computer Aided Design of Integrated Circuits and Systems. vol. 27, No. 8, pp. 1479-1492, 2008.

[10] K.Puttaswamy and Gabriel.H. Loh, " Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High performance 3D-Integrated Processor", HPCA'07: International Symposium on High-Performance Computer Architecture, pp. 193-204, 2007.

[11] Gabriel H. Loh, "Extending the Effectivenes of 3D –Stacked DRAM Caches with an Adaptive Multi-Queue Policy", MICRO'09: International Symposium on Microarchitecture , pp. 201-212, 2009.

[12] S. Thoziyoor, J. Ho Ahn, A. Monchiero, J. B. Brockman and N. P. Jouppi, "A Comprehensive Memory Modeling Tool and its Application to the Design and Analysis of Future Memory Hierarchies", ISCA08: International Symposium on Computer Architecture, pp. 51-62 , 2008.

[13] Compaq Computer Corporation. Alpha 21264 Microprocessor Hardware Reference Manual, 1999.

[14] K. Puttaswamy and G. Loh, "Implementing Register Files for High-Performance Microprocessors in a Die-Stacked (3D) Technology", ISVLSI' 06: International Symposium on VLSI, pp. 384-389, 2006.

[15] D. Ernst, T. Austin, "Efficient dynamic scheduling through tag elimination", ISCA'02: International Symposium on Computer Architecture, pp. 37-46, 2002.

[16] K. Puttaswamy and G. Loh. "Dynamic Instruction Schedulers in a 3-Dimensional Integration Technology", GLSVLSI'06: ACM/IEEE Great Lakes Symposium on VLSI, pp. 153-158, 2006.

[17] G. H. Loh, "A Modular 3D Processor for Flexible Product Design and Technology Migration", Proceedings of the 5th ACM conference on Computing Frontiers, pp. 159-170, 2008.

[18] D.H. Albonesi et al., "Dynamically Tuning Processor Resources with Adaptive Processing", IEEE Computer Society, vol 36, Issue 12, pp. 49-58,2003.

[19] Sheng Li et al., "McPat: An Integrated Power, Area and Timing Modeling Framework for Multicore and Manycore Architectures", MICRO'09: International Symposium on Microarchitecture, pp. 469-480, 2009.

[20] R. Desikan, D. Burger, and S.W. Keckler, " Measuring Experimental Error in Microprocessor Simulation", ISCA'01: International Symposium on Computer Architecture, pp. 266-277, 2001.

[21] Y. Xie, G. H. Loh, B. Black, K. Bernstein, "Design Space Exploration for 3D Architectures", ACM Journal on Emerging Technologies in Computing Systems, vol. 2, No.2, pp. 65-103, 2006.