# High Performance Non-blocking Switch Design in 3D Die-Stacking Technology

Dean L. Lewis Sudhakar Yalamanchili Hsien-Hsin S. Lee

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
{dean, sudha, leehs}@ece.gatech.edu

## ABSTRACT

Die stacking is a promising new technology that enables integration of devices in the third dimension. It allows the stacking of multiple active layers directly on top of one another with short, dense die-to-die vias providing communication. Previous work has shown significant benefits at all design targets, from stacking memory on logic to partitioning individual architectural units across multiple layers. Many high-speed processor units—ALUs, register files, caches, and instruction schedulers—have all been designed in 3D, achieving significant, simultaneous power savings and performance boosts. Other work has looked at the implementation of network-on-chip in a die stack but restricted the focus to planar designs of the various unit (processors, routers, etc.). This work follows up on these two research areas to explore the 3D design of router components, specifically the crossbar. We examine the implementation of a crossbar and two multistage interconnect networks to determine the potential benefits of 3D implementations. Compared to equivalent planar designs, we achieve a maximum delay reduction of 26% and maximum power savings of 24%.

## Keywords

3D Integration, Die-Stacking, NoC, Crossbar Design, MIN Design

## 1. INTRODUCTION

In a continuing effort to keep up with the relentless march of Moore's law, processor designers keep pushing the limits of technology further and further. Unfortunately, each push inevitably costs more than the last—more money and more time. To make matters worse, the returns from each push are steadily diminishing. Each new technology generation consumes more power, becomes less reliable, and fails to achieve an ideal performance improvement. Consequently, researchers continue to seek out innovative new technologies orthogonal to technology shrinks. 3D integration—also known as *die stacking*—is a very promising technology that enables IC design in the third dimension, making an entire system on a single die possible and continuing the scaling trajectory predicted by Moore's Law for a few more generations [5].

For 3D-integrated microprocessors, prior research thrusts proposed and studied several methods for partitioning their functions [3, 4, 8, 15, 16, 18, 19, 12]. These partitioning schemes range from simple die-stacking of memory chips on a processor to partitioning microarchitectural blocks or even a single functional unit (e.g. an adder) across different die layers. In this paper, we extend this body of work by focusing on the design of routers using 3D technology. Specifically, we take a look at potential implementations of crossbars in 3D and report on their respective power and performance benefits.

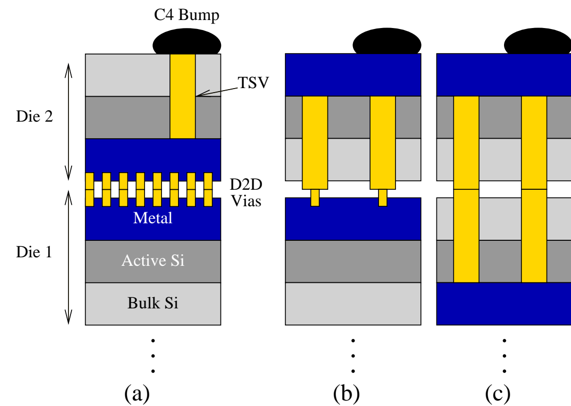



**Figure 1: Three die stacks, each comprised of two layers using three possible bond styles: (a) face-to-face, (b) face-to-back, and (c) back-to-back**

The rest of this paper is organized as follows. Section 2 gives an overview of 3D integration and the range of potential applications. Section 3 will look at various planar implementations of crossbars. Section 4 will present the redesigned 3D versions of these crossbars. Section 5 presents our experimental setup and results. Section 6 concludes the paper with a summary and discussion of results.

## 2. OVERVIEW OF 3D-IC TECHNOLOGY

3D integration is an emerging technology that allows semiconductor die to be bonded together to form a tightly integrated stack. Opening design to the third dimension provides several advantages. First, it enables the integration of heterogeneous components such as logic and DRAM memory [3] or analog and digital circuits [2]. Secondly, it increases routability [14]. Last but not least, it can substantially reduce wire length, which contributes both to long communication latency and to high power consumption. Recent work in this field has already demonstrated significant improvements in both performance and power consumption [17] and lead to other interesting applications, such as online profiling [10], flexible snap-on accelerator [23], network-in-memory [8], and programmable test [9]. Even greater returns are expected as researchers further explore the opportunities afforded.

### 2.1 3D Die Bonding

Figure 1 shows simple two-layer die stacks. The two die communicate through an array of die-to-die (d2d) vias, which come in two

flavors: faceside and backside. Faceside vias are manufactured on top of the metal layers with size and pitch on the order of a few hundred nanometers [21]. Backside vias—also called *Through Silicon Vias (TSVs)*—are etched through the active and bulk silicon with size and pitch on the order of microns. Exposing backside vias requires that the die be thinned from several hundred microns to only a few tens of microns. With these vias exposed, the die can be bonded to the other die in the stack [13]. There are three possible types of bonding: face-to-face (Figure 1(a)), face-to-back (Figure 1(b)), and back-to-back (Figure 1(c)). Face-to-face is the superior interface because it provides a significantly higher via density. However, back-to-back and face-to-back interfaces are required to stack beyond two die layers.

Utilizing these different interface options, designers continue to push further into the third dimension. Some embedded applications already utilize a die-stack with eight layers [20]. Given the disparity between faceside and backside via densities, face-to-face bonds are more appropriate for small granularity partitions while back-to-back bonds are better suited to coarse-grained partitions. When the stack is complete, the requisite C4 solder bumps can be placed on the TSVs of a backside layer (as shown in Figure 1) or on the top metal layer of a faceside layer (just as in planar designs).

## 2.2 3D Partitioning Granularity

3D die stacking technology may be used to partition a design at three general levels of granularity. The coarsest level is the technology level. Disparate technologies like high-speed CMOS and high-density DRAM both have their own dedicated and highly-optimized manufacturing processes. Many problems arise when attempting to integrate such technologies onto a single die, requiring sophisticated manufacturing tricks to achieve economically viable integration quality [11]. Die stacking allows each technology to be manufactured on its own layer in its own process. After each layer is manufactured, a separate integration process bonds these layers together. The result is the best of both worlds: each layer is manufactured at the highest possible quality level and, simultaneously, the two technologies are tightly integrated. This improves both the performance of the system and the form factor.

The next finer level of partitioning is the architectural level. Unlike technology partitioning, both layers are manufactured using the same process. The goal of architectural partitioning is to spread the functional blocks of a design such as a microprocessor across the available layers in such a way as to minimize the length of the interconnect buses, while maintaining each functional block as a 2D module. By reducing bus length, the resistance and capacitance on these buses are reduced, consequently reducing power consumption and improving performance. The architectural partitioning scheme makes much better use of the large number of d2d vias available than the technology partitioning.

The finest partitioning granularity is the circuit level. Here, the transistors that make up a functional block may exist on different layers. Circuit partitioning has its own levels of granularity. At one extreme, blocks are simply split along logical boundaries into sub-blocks (e.g. a design could place half the banks of a cache on one layer and the other half on a different layer—so called bank-stacking [8, 15]). At the other extreme, individual circuits are split across the layers [17, 19, 12] (e.g. in a register file, read ports and write ports may be spread across different layers, connected to the actual memory cell through d2d vias; this is known as port-splitting [17]). This granularity best utilizes the available d2d vias and thus shows the best power and performance improvements. Nonetheless, such paritioning styles will make chip testing (e.g. isolating known good die) a huge challenge [6, 7] .

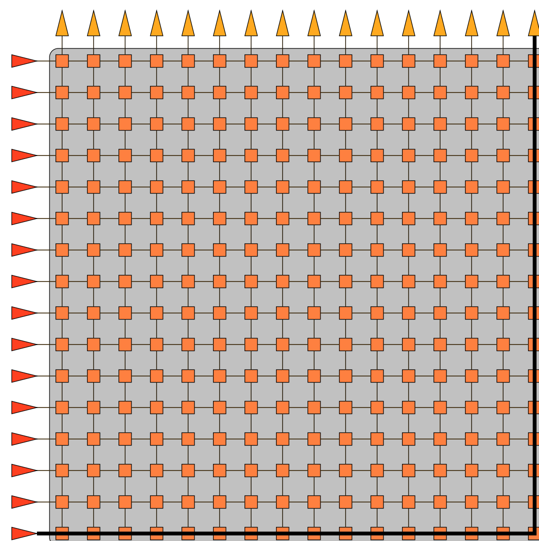


**Figure 2: A traditional 16-by-16 crossbar.**

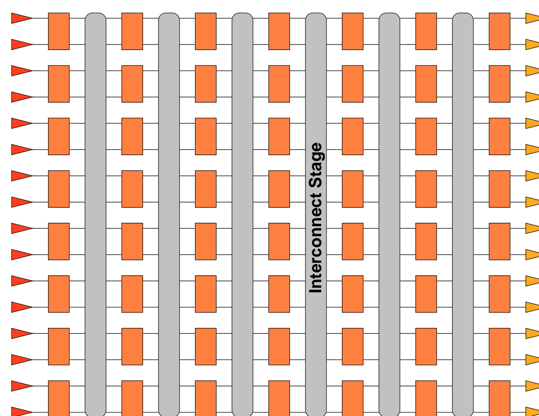



**Figure 3: The general topology 16-by-16 MIN network.**

# 3. NON-BLOCKING SWITCH DESIGN

Non-blocking switches are an essential component of any router. One of the most popular of these is the crossbar. The crossbar's high degree of connectivity allows for a large number of simultaneous connections to keep data moving through the network. Unfortunately, this connectivity comes at a high cost. Crossbars utilize a large number of switches and a large number of wires, which translates into high power consumption, large size, and low operational frequencies.

A traditional $n$-by-$n$ crossbar is shown in Figure 2, a very simple and clean design with short and easy wiring. The cost, however, is in the exceptionally large number of switches required. Data must traverse $2n - 1$ switches on the longest path to cross the crossbar, a very long path indeed. Thus, designers have come up with alternative circuits that can significantly reduce the number of switches required.

The key feature of the crossbar is that it is a *strictly non-blocking network*; that is, any free input port can be connected to any free output port without changing existing input/output pairs. However, strickly non-blocking may be too cumbersome a design constraint. Thus we also consider *multistage interconnection networks* (MINs), specifically rearrangable MINs like the Benes network.[1] A rearrangeable network allows all possible connection pairs but may re-

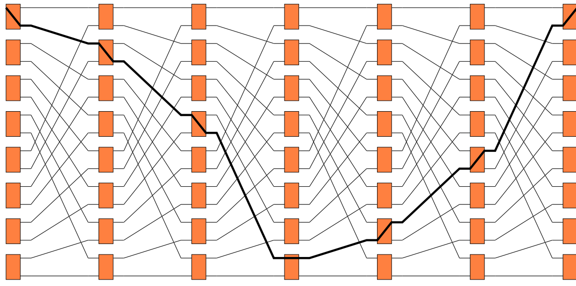


**Figure 4: A 16-by-16 perfect shuffle network.**

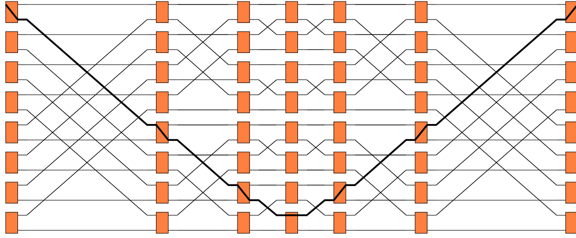
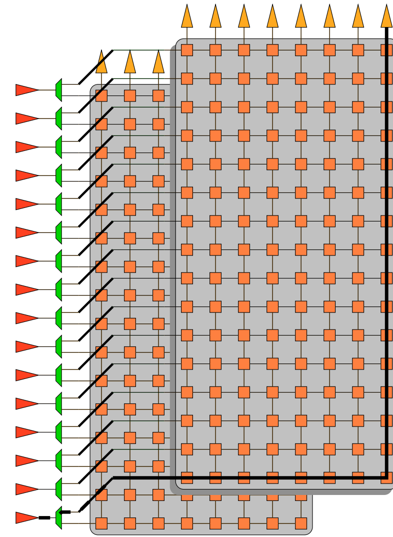


**Figure 5: A 16-by-16 butterfly network. Note the significant difference in wiring complexity, depending on the stage.**

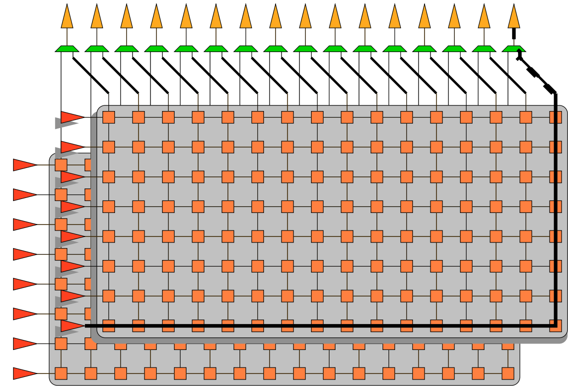
quire existing connections to change to accomodate new connections. The general structure of a rearrangable MIN (hereafter referred only as a MIN) is shown in Figure 3. It is essentially a series of switching and interconnection stages. An $n$-by-$n$ MIN network will require $2\log_2 n - 1$ switch stages (using 2-by-2 switches) and $2\log_2 n - 2$ shuffling stages. In the switching stages, the two inputs at each switch can be passed directly to their respective outputs, or they can be swapped to the opposite outputs. In each interconnetion stage, wires are used to pass signals between different switches. A MIN network is rearrangeable because it allows every input one unique path to any port on the middle switch stage and one unique path from each middle switch stage port to each output. This results in $n$ paths between each input and output, which is sufficient for routing any and all non-interfering connection patterns.

The trick to MIN design lays in the interconnect. These stages must be designed in such a way as to allow any input to reach every switch in the middle stage. There are plenty of permutation algorithms that accomplish this. We have selected two representative algorithms for 3D implementation. The first is the perfect shuffle, shown in Figure 4. In the perfect shuffle, the first $\frac{n}{2}$ ports are interleaved with the second $\frac{n}{2}$ ports, as if a deck of $n$ cards were perfectly shuffled. This shuffle is uniform, meaning each wiring stage looks identical to every other. This means equal delay and equal wiring complexity, which may be desirable.

The other design is the butterfly network, shown in Figure 5. A butterfly network acts similarly to a barrel shifter in an ALU unit, allowing data to change from source address to destination address by one, then by two, by four, and so on up to $\frac{n}{2}$. This interconnect is non-uniform, which means the delay and wiring complexity changes from stage to stage. Note that routing on the most significant bit (leftmost and rightmost stages) involves significant wiring, which equals significant power and delay. Routing on the least significant bit (innermost stages), on the other hand, is simple and quick. Ideally, we would like to avoid the cost of the MSB interconnect and utilize more LSB-like wiring, though there is no way to accomplish this in a planar design.



(a)



(b)

**Figure 6: Two 3D implementations of a 3D crossbar. (a) shows a row-split (i.e., the row is folded in two, placing half the columns on the top layer) 3D crossbar. (b) shows the opposite fold, a column-split 3D crossbar. Multiplexers (shown in green) have been added to shorten the critical paths.**

## 4. 3D NON-BLOCKING SWITCHES

The goal of a 3D implementation is to reduce as much as possible the length of the critical path through the circuit. There is a great variety of ways to create a 3D implementation. The simplest is to fold the circuit in half. Unfortunately, this reduces only the footprint, not the wiring or area consumed. It is much better to attempt to cut internal wires and bring the actual transistors closer together.

A crossbar, unfortunately, has little wiring to target for cutting. However, it does have an extremely long critical path (from the bottom-most input to the right-most output as shown in Figure 2). Thus, we choose to fold the crossbar and then add extra switches in order to reduce the critical path. Figure 6 shows two 3D crossbars, each folded along a different axis. Multiplexers have been added to cut the critical path in half in one dimension. Thus, in both cases, the critical path is three-quarters as long as in the planar design, plus the delay of one multiplexer. This design can be extended to four layers by simply combining these two splits.

Unlike the crossbar, the perfect shuffle, has plenty of wiring we can reduce. A careful examination of the wiring stages reveals a pattern

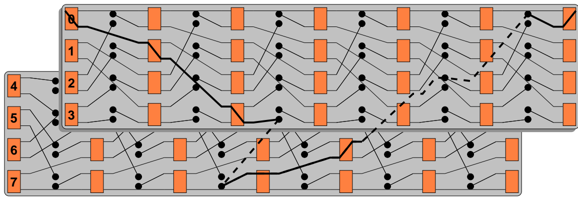



**Figure 7: A 16-by-16 bit-partitioned 3D perfect shuffle network. The dots represent via connections. The numbers represent the three most significant bits of the port addresses feeding that switch, i.e. switch 0 receives ports 0 (0000) and 1 (0001).**

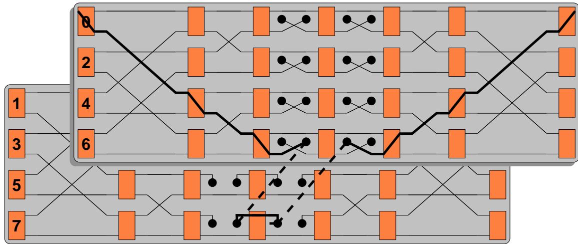



**Figure 8: A 16-by-16 bit-partitioned 3D butterfly network. The dots represent via connections. The numbers represent the three most significant bits of the port addresses feeding that switch, i.e. switch 0 receives ports 0 (0000) and 1 (0001).**

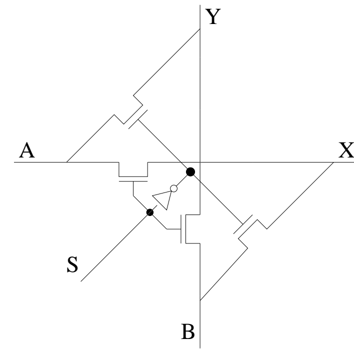



**Figure 9: The schematic of a one bit switch.**

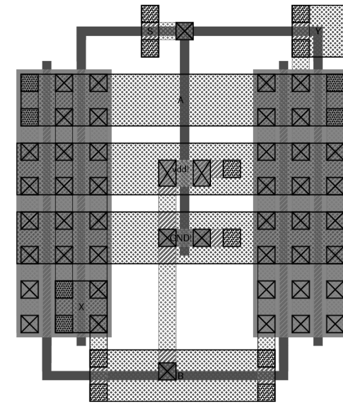


**Figure 10: The layout of a one bit switch.**

we can exploit. The longest paths are represented by switch `four` in a given stage communicating with switches `zero` and `one` in the next stage. Similarly, switch `three` communicates with `six` and `seven`. So switches `zero` and `four` share destinations, as do switches `three` and `seven`. Thus, for the 3D implementation, we split the crossbar in half and slide the bottom half under the top half, as shown in Figure 7. This stacks `zero` on top of `four` and `three` on top of `seven`, greatly reducing the longest stage wires. Wiring from stages `two` and `five` are also reduced. Most importantly, though, the wiring complexity (i.e. the number of crossing wires) has been reduced from seven to three.

The butterfly network, on the other hand, has a disproportionately large wiring cost in the two outermost stages. One potential design approach would be to directly target these stages for 3D design. However, this is not the most effective solution because we would be limiting the potential benefit to only those stages. Instead, we shrink these stages indirectly by targeting the innermost stages (Figure 8), which benefits nearly every stage. We apply the bit-partitioning method, first described in [15], to the innermost stages. Basically, by placing vias in these two stages and moving every other switch to the second layer, we have moved every other connection in all other stages to the second layer as well. This cuts the wiring density of each wiring stage in half as well, greatly reducing the wiring length, and thus power and delay, required at each stage. This is a significant improvement over the planar case.

Additionally, this design scales well to more layers. At two layers, as shown in Figure 5, we partition only the two innermost stages. With four layers, we can partition the next two innermost layers. With eight, the next two, and so on and so on. With each doubling in die count, the wiring complexity at the outer stages is cut in half. Simultaneously however, the complexity of the via network in the middle grows. Therefore, there is a trade-off point where more layers begin to hinder the design. This point, however, will depend on the size of the switch being implemented, and thus will differ from design to design.

## 5. EXPERIMENTS

For our experiments, we perform circuit-level HSPICE simulations of the critical path assuming the worst-case switching. The simulations are based on 130nm level 49 transistor models and 130nm wire parameters published from Intel [22]. We estimate the size of vias at $10\mu m * 1.7\mu m * 1.7\mu m$ based on 3D die-stacking technology developed at Tezzaron [21]. The switches are all one-bit wide (as shown in Figure 9). To properly estimate the switch size, the switch was laid out in 130nm SCMOS technology (Figure 10). A stick diagram was extracted from this layout, and then the stick diagram was scaled to Intel's 130nm technology rules.

The switch shown in Figure 10 is slightly overdesigned. First, B, the second input, has wiring tracks available for both bottom and left connections, and Y, the second output, has similarly available tracks for both top and right connections. This is to accomodate both the crossbars (using the bottom and top connections) and the MINs (using the left and right connections). In a real implemention, only one pair of connections would be made; both are included here to show either configuration is possible without significant changes to the switch design. In the experiments, we assume only the minimum wiring required for the given crossbar type. Second, while MINs require a full two-by-two switch to function, crossbars do not. However, to keep the experiments simple, we used the full switch in the crossbars as well. This will increase area, delay, and power compared to a more optimized design, but it should not significantly affect the relative advantages of 3D design.

| Design | Area ($\mu m^2$) | Delay (ns) | Energy (pJ) |
|---|---|---|---|
| Planar Crossbar | 3772 | 2.882 | 0.347 |
| Planar Perfect Shuffle | 1363 | 0.739 | 0.162 |
| Planar Butterfly Network | 1139 | 0.734 | 0.090 |
| 3D Row-split Crossbar | 3823 | 2.202 | 0.277 |
| 3D Column-split Crossbar | 3841 | 2.131 | 0.263 |
| 3D Perfect Shuffle | 1161 | 0.688 | 0.146 |
| 3D Butterfly Network | 1004 | 0.675 | 0.082 |

**Table 1: Reported area is for the entire design—bottom-layer area plus top-layer area for 3D designs. Delay and power are reported for the critical path through each design.**

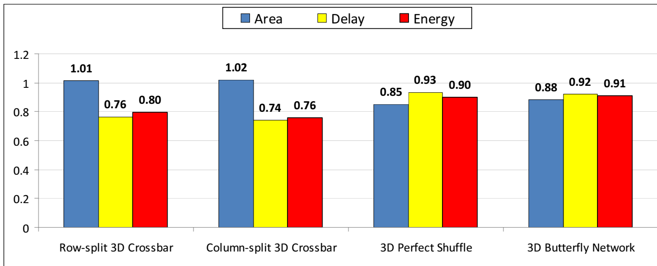



**Figure 11: Area, delay, and power results for the 3D crossbar designs, each normalized to its equivalent planar design.**

Results for area, delay, and power are given in Table 1. For 3D designs, the reported area is the sum of the area used in the bottom layer and the area used in the top layer. Because of this, both the row-split and column-split 3D crossbars are slightly larger than the planar crossbar, thanks to the additional multiplexers. The footprint of each 3D design is half of the reported area.

For comparison, results for the 3D designs are shown in Figure 11. The results for each design have been normalized to the equivalent planar design. There are a couple key points to note. First, while both 3D true crossbars have slightly higher area than the planar version, both are significantly faster (24% for row-split and 26% for column-split) and, simultaneously, less power-hungry (20% and 24% respectively). The slight difference between the two designs is due to the fact that the switches are not square, and thus the cost of a horizontal connection is different from the cost of a vertical connection.

Second, the MIN designs show improvement in every metric. This showcases the power of 3D design to untangle large interconnects and really shorten costly wiring. Note that, though the butterfly network uses many fewer TSVs than the perfect shuffle, it achieves similar improvements over its planar equivalent. This demonstrates the importance of strategic TSV placement; depending on the circuit, a few well-placed TSVs can be just as effective as many.

# 6. CONCLUSION

Previous work has explored the design of 3D functional units in processor cores design. We have taken these design techniques and applied them to another critical, high-speed component: the crossbar unit of a router. Our results suggest two different routes for future crossbar design. For small-scale networks (i.e. networks-on-chip), true 3D crossbars are great options because they can reduce power and delay even at very small unit sizes. For large-scale networks (i.e. rack-based) where a router may be an entire chip and crossbars are impractical, a MIN is the better choice, and 3D implementations of these crossbars show promise for significantly reducing power consumption and delay. Iin each direction networking is moving, 3D can help.

# 7. ACKNOWLEDGMENT

This research is supported in part by the C2S2 center of the SRC's Focus Center Research Program and an NSF grant CCF-0811738.

# 8. REFERENCES

[1] V. E. Benes. Mathematical theory of connecting networks and telephone traffic. *Academic Press*, 1965.

[2] S. Bhansali, G. Chapmann, E. Friedman, Y. Ismail, P. Mukund, D. Tebbe, and V. Jain. 3-d heterogeneous sensor system on a chip for defense and security applications. In *Proceedings of SPIE Volume 5417*, pages 413–424, 2004.

[3] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G. H. Loh, D. McCauley, P. Morrow, D. W. Nelson, D. Pantuso, P. Reed, J. Rupley, S. Shankar, J. Shen, and C. Webb. Die Stacking (3D) Microarchitecture. In *Proceedings of the 39th International Symposium on Microarchitecture*, 2006.

[4] B. Black, D. Nelson, C. Webb, and N. Samra. 3D Processing Technology and Its Impact on iA32 Microprocessors. In *Proceedings of the 22nd International Conference on Computer Design*, pages 316–318, 2003.

[5] S. Gupta, M. Hilbert, S. Hong, and R. Patti. Techniques for producing 3d ics with high-density interconnect. In *VMIC '04: Proceedings of the 21st International VLSI Multilevel Interconnection Conference*, Waikoloa Beach, HI, USA, 2004.

[6] H.-H. S. Lee and K. Chakrabarty. Test Challenges for 3D Integrated Circuits. *To appear in IEEE Design and Test of Computers, Special Issue on 3D IC Design and Test*, Sep/Oct 2009.

[7] D. L. Lewis and H.-H. S. Lee. Testing Circuit-Partitioned 3D IC Designs. In *IEEE Computer Society Annual Symposium on VLSI*, 2009.

[8] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, N. Vijaykrishnan, and M. Kandemir. Design and Management of 3D Chip Multiprocessors using Network-in-Memory. In *Proceedings of the International Symposium on Computer Architecture*, 2006.

[9] J. Li, S. Ghosh, and K. Roy. A generic and reconfigurable test paradigm using low-cost integrated poly-si tfts. In *IEEE International Test Conference*, October 2007.

[10] S. Mysore, B. Agrawal, N. Srivastava, S.-C. Lin, K. Banerjee, and T. Sherwood. Introspective 3D Chips. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, 2006.

[11] Y. Nunomura and N. Manijikian. M32r/d-integrating dram and microprocessor. *IEEE MICRO*, 17(6):40–48, 1997.

[12] D. Park, S. Eachempati, R. Das, A. K. Mishra, Y. Xie, N. Vijaykrishnan, and C. R. Das. MIRA: A Multi-layered On-Chip Interconnect Router Architecture. In *Proceedings of the 35th International Symposium on Computer Architecture*, pages 251–261, 2008.

[13] R. Patti, M. Hilbert, S. Gupta, and S. Hong. Techniques for producing three dimensional integrated circuits with high density interconnect. In *International VLSI Multilevel Interconnection Conference*, 2004.

[14] V. Pavlidis and E. Friedman. 3-d topologies for networks-on-chip. In *International SOC Conference*, pages 285–288, 2006.

[15] K. Puttaswamy and G. H. Loh. Implementing Caches in a 3D Technology for High Performance Processors. In *Proceedings of the International Conference on Computer Design*, 2005.

[16] K. Puttaswamy and G. H. Loh. Dynamic Instruction Schedulers in a 3-Dimensional Integration Technology. In *Proceedings of the ACM/IEEE Great Lakes Symposium on VLSI*, 2006.

[17] K. Puttaswamy and G. H. Loh. Implementing register files for high-performance microprocessors in a die-stacked (3d) technology. In *ISVLSI '06: Proceedings of the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures*, page 384, Washington, DC, USA, 2006. IEEE Computer Society.

[18] K. Puttaswamy and G. H. Loh. The Impact of 3-Dimensional Integration on the Design of Arithmetic Units. In *Proceedings of the International Symposium on Circuits and Systems*, 2006.

[19] K. Puttaswamy and G. H. Loh. Thermal Herding: Microarchitecture Techniques for Controlling Hotspots in High-Performance

3D-Integrated Processors. In *Proceedings of the 13th International Symposium on High-Performance Computer Architecture*, pages 193–204, 2007.

[20] Samsung. http://www.samsung.com/presscenter/pressrelease/pressrelease.asp?seq=20060413_0000246668. 2006.

[21] Tezzaron. http://www.tezzaron.com/technology/fastack.htm. 2006.

[22] S. Thompson, M. Alavi, M. Hussein, P. Jacob, C. Kenyon, P. Moon, M. Prince, S. Sivakumar, S. Tyagi, and M. Bohr. 130nm logic technology featuring 60nm transistors, low-k dielectrics, and cu interconnects. *Intel Technology Journal*, 6(2), May 2002.

[23] D. H. Woo, H.-H. S. Lee, J. B. Fryman, A. D. Knies, and M. Eng. POD: A 3D-Integrated Broad-Purpose Acceleration Layer. *IEEE Micro*, 28(4):28–40, 2008.