A Hexagonal Shaped Processor and Interconnect Topology for Tightly-tiled Many-Core Architecture

Zhibin Xiao and Bevan Baas

Department of Electrical and Computer Engineering, University of California, Davis

Abstract— **2-Dimensional meshes are the most commonly used Network-on-Chip (NoC) topology for on-chip communication in many-core processor arrays due to their low complexity and excellent match to rectangular processor tiles. However, 2D meshes may incur local traffic congestion for applications with significant levels of traffic with non-neighboring cores, resulting in long latencies and high power consumption. In this paper, we propose an 8-neighbor mesh topology and a 6-neighbor topology with hexagonal-shaped processor tiles. A 16-bit DSP processor and the corresponding processor arrays are implemented in all three topologies. The hexagonal processor tile and arrays of tiles are laid out using industry-standard CAD tools and automatic place and route flow without full-custom design, and result in DRC-clean and LVS-clean layout. A 1080p H.264/AVC residual video encoder and a 54 Mbps 802.11a/11g OFDM wireless LAN baseband receiver are mapped onto all topologies. The 6-neighbor hexagonal grid topology incurs a 2.9% area increase per tile compared to the 4-neighbor 2D mesh, but its much more effective inter-processor interconnect yields an average total application area reduction of 21%, an average power reduction of 17%, and a total application inter-processor communication distance reduction of 19%.**

I. INTRODUCTION

Tiled architectures that integrate two or more independent processor cores are called multi-core processors. Manufactures typically integrate multi-core processors into a single integrated circuit die (known as chip multiprocessors or CMP). CMPs that integrate tens, hundreds, or thousands of cores per die are called *many-core* chips and those that utilize scalable interconnects and avoid long global wires will attain higher performance [1].

NoCs are used to connect large numbers of processors in many-core processor architecture because they perform better than less scalable methods such as global shared buses. Among all NoC design parameters, NoC topologies define how nodes are placed and connected and greatly affect the performance, energy efficiency, and circuit area of many-core processor arrays. Due to its simplicity and the fact that processor tiles are traditionally square or rectangular, 2D mesh is mostly used for existing on-chip networks. However, efficiently mapping applications can be a challenge for cases that require communication between processors that are not adjacent on the 2D mesh. This condition could require processors to act as routing processors for static interconnection architectures, and intermediate routers for dynamic router-based NoCs. The power consumption and communication latency also increase as the number of routing processors or routers between two communicating cores increase.

For many applications mapped onto homogeneous chip multiprocessors, communication within processors is often largely localized [2], which may result in local mapping congestion. An increase of local connectivity can ease such congestion, which results in application mappings with smaller application area and lower power consumption. This motivates us to propose new topologies with increased local connectivity while keeping much of the simplicity of a mesh-based topology.

The main contributions of this paper can be summarized as three points. First, we have proposed a 6-neighbor topology with hexagonal-shaped processor tiles and a 8-neighbor mesh topology, which are compared to the common 4-neighbor 2D mesh topology. Second, commonly available commercial CAD tools are used to implement tiled CMPs for all three topologies. Three processors including a hexagonal-shaped processor tile and their corresponding many-core processor arrays are physically implemented in 65 nm CMOS and are DRC and LVS clean. Third, a complete functional H.264/AVC residual encoder and an 802.11a baseband receiver are mapped onto all three topologies for realistic comparisons.

The remainder of this paper is organized as follows. Section II describes the related work. Section III presents the proposed inter-processor communication topologies. Section IV shows the mapping of two complex applications to all discussed topologies. In section V, the physical design of the hexagonal-shaped processor tiles is presented. Section VI presents the chip implementation results and section VII concludes this paper.

II. RELATED WORK

Many topologies have been used for on-chip inter-processor communication, such as buses, meshes, tori, binary trees, octagons, hierarchical buses and custom topologies for specific applications. The low complexity 2D mesh has been used by most fabricated many-core systems including RAW [3], AsAP [4], TILE64 [5], AsAP2 [6] and Intel 48-core Single-Chip Cloud Computer (SCC) [7].

Allen studies the two-dimensional regular processor arrays which are geometrically defined based on nearest-neighbor connections and space-filling properties [8]. He theoretically proves the hexagonal array is the most efficient topology in emulating other topologies by analyzing the geometric characteristics. Chen et al. theoretically explored the addressing, routing and broadcasting in hexagonal mesh multiprocessors [9].

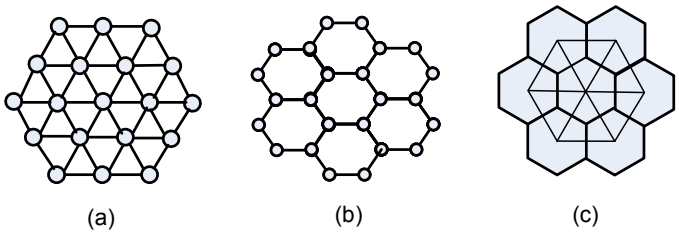

Fig. 1. Examples of three hexagonal networks (a) a 6-neighbor off-chip hexagonal network; (b) a 3-neighbor on-chip honeycomb network [12]; (c) the proposed 6-neighbor on-chip hexagonal grid network.

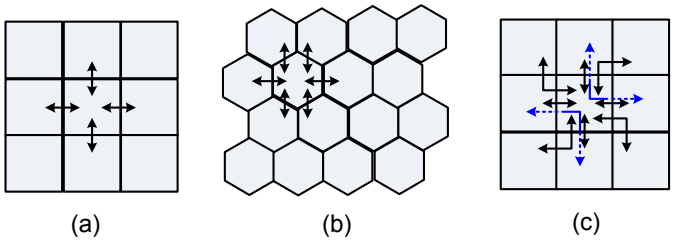

Fig. 2. The three inter-processor communication topologies considered in this work: (a) baseline 4-neighbor mesh (b) 6-neighbor hexagonal tile and interconnect, and (c) 8-neighbor mesh.

TABLE I

LINK LENGTH FOR THE THREE STUDIED TOPOLOGIES WITH THE AREA OF EACH PROCESSOR TILE EQUAL TO ONE UNIT OF LENGTH SQUARED.

Topology	**Nearest-neighbor Link**		**Longer Link**	
	Number	Length	Number	Length
4-neighbor mesh	4	1.00	–	–
6-neighbor hex grid	6	1.07	–	–
8-neighbor mesh	4	1.00	4	1.41

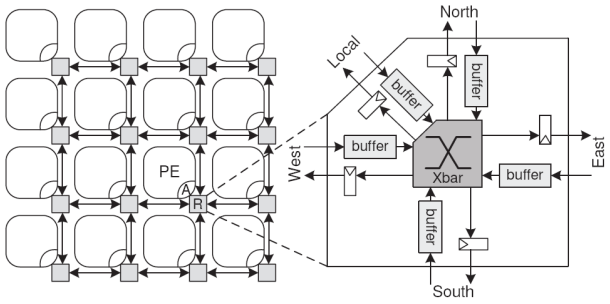

Fig. 3. A 2D mesh processor array connected by a dynamic five-port routers each with one port connected to the processor core.

Decayeux and Seme proposed a 3D hexagonal network as an extension of 2D hexagonal networks [10]. Their work focuses on off-chip 6-neighbor hexagonal network where each node is located at the vertex of the network as shown in Figure 1(a). Stojmenovic proposed efficient coordinate system and routing algorithms for the 3-neighbor honeycomb mesh networks as shown in Figure 1(b) [11]. Compared to previous work, we have designed a hexagonal-shaped processor that can be tiled together as a hexagonal mesh for on-chip inter-processor communication as shown in Figure 1(c). The advantages of hexagonal-shaped processor topology are demonstrated by real-world application mappings and physical implementations of a fully functional many-core processor array.

III. PROCESSOR SHAPES AND TOPOLOGIES

As shown in Fig. 2, three different topologies are studied and the well-known 4-neighbor mesh is used as the baseline topology for comparison as shown in Figure 2(a).

Figure 2(b) shows a 6-neighbor processor array using hexagonal-shaped processor tiles. The processor center-to-center distance is $\sqrt{3} * w$ if the length of the hexagon edge is w. The hexagonal grid is commonly used in mobile wireless networks due to its desirable feature of approximating circular antenna radiation patterns and its optimal characteristic of six nearest neighbors. The symmetry and space-filling property make the hexagonal-shaped processor tile an attractive design option for many-core processor arrays.

Due to limitations of current wafer sawing machines, chips on round wafers are traditionally square or rectangular. In fact, the opportunities and limitations of non-rectangular processors on a chip are analogous to non-rectangular chips on a wafer. For the case of a rectangular chip composed of hexagonal-shaped processors, there are areas on the periphery of the chip in which processors can not be placed. If the hexagonal processor array size is larger than 30 x 30, this area overhead becomes less than 2.7% of the total chip area. In practice, this area could be filled with other chip components such as decoupling capacitors, or portions of hardware accelerators, memory modules, I/O circuits or power conversion circuits.

Another logical extension of the 2D mesh is to include four diagonal processors in an 8-neighbor arrangement as shown in Figure 2(c) where each rect tile can directly communicate with 8 neighbors. This approach has increased routing congestion in the tile corners due to the four (uni-directional) links that pass through each corner (the dashed lines in Figure 2(c)).

The center-to-center distance can be used to represent the communication link length between two "touched" processors. Table I shows the number of different types of communication links and the corresponding link length for the three topologies. The area of a processor tile is assumed to be one squared unit. As shown in Table I, the 4-neighbor mesh and 6-neighbor hex grid have only one type of communication link due to equal center-to-center tile distance. The 8-neighbor mesh topology has two types of links.

IV. APPLICATION MAPPING

A. Target Interconnect Architecture

Fig. 3 shows the inter-processor communication in a typical 2D mesh processor array using dynamic routers. As the diagram shows, the processor array is connected by 5-port routers and the communication logic includes five buffers and one 5x5 crossbar. There might be more control logic to support the communication flow control which is not drawn. The static circuit-switch interconnection has smaller area, lower power dissipation and lower complexity than dynamic router

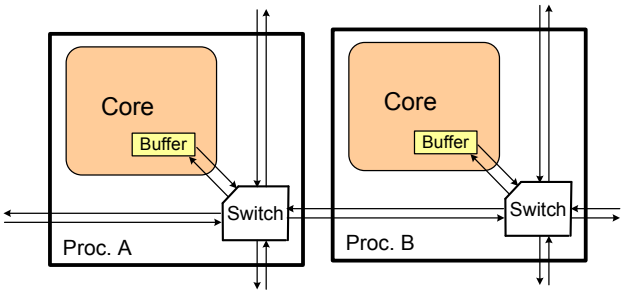

Fig. 4. A 2D mesh processor array connected by circuit switches each with four nearest-neighbor inter-processor communication links and one port connected to the processor core.

interconnection while trading off routing flexibility [13]. Fig. 4 shows another 2D mesh array connected by circuit switches each with four nearest-neighbor interconnection links and one port connected to the processor core. The circuit switch communication logic has only one buffer and one 4x1 crossbar. The long distance communication is performed by software in the intermediate processors. In this work, we use the static configurable circuit switch architecture which is suitable for applications with steady communication patterns. We also extend the architecture in Fig. 4 by adding one more port to the processor core due to the fact that processors normally have a two-operand instruction format. Thus, the processor can read two words from two buffers in one instruction at the same time.

B. Application Mapping Results

Parallel programming on fine-grained many-core systems includes two steps: 1) Partitioning the algorithms at a fine-grained level; 2) Mapping the tasks to the nodes of the processor array and connecting the nodes with available links defined by the topology [14]. The algorithm partitioning and mapping can be conducted either manually or automatically. The partitioning problem is out of the scope of this paper and the details are not discussed. This section focuses on the second step – application mapping.

Based on the two-port circuit switch architecture, two complete applications including an H.264/AVC residual encoder and an 802.11a receiver are manually mapped onto all three topologies which differ in the number of links among neighboring processor tiles.

Figure 5 depicts two task graphs of the benchmark applications, where each node represents one task which can be implemented in one processor and each edge represents one physical link between two processor nodes. Figure 5(a) shows a 22-node task graph of an H.264/AVC residual baseline encoder composed of integer transform, quantization and context-adaptive and variable length coding (CAVLC) functions [14]. The H.264/AVC encoder is a memory-intensive application which requires an additional shared memory module as shown in the task graph. Figure 5(b) shows a 22-node task graph of a complete 802.11a WLAN baseband receiver which is computation-intensive requiring two dedicated hardware

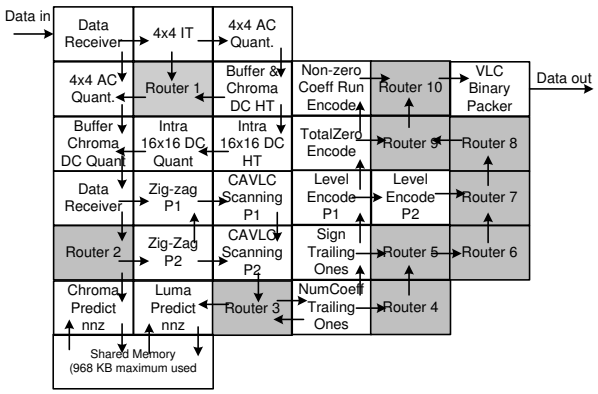

Fig. 6. An H.264/AVC video residual encoder mapped on a processor array with 4-neighbor 2D mesh topology.

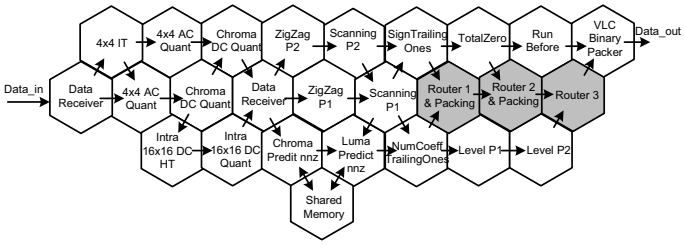

Fig. 7. An H.264/AVC residual video encoder mapped on a processor array with 6-neighbor hex topology.

accelerators: Viterbi decoder and FFT. The complete receiver includes necessary practical features such as frame detection, timing synchronization, carrier frequency offset (CFO) estimation and compensation, and channel estimation and equalization [15].

Figure 6 shows an example mapping of the H.264/AVC residual encoder capable of 1080p HDTV encoding at 30 frames per second on the baseline 4-neighbor mesh that uses 32 processors plus one shared memory. The 4-neighbor mesh is inefficient in handling a complex application like H.264/AVC encoding. A total of 10 processors are used solely for routing data which accounts for 31% of the total application area. Figure 7 shows a possible 25-processor mapping on the proposed 6-neighbor hex grid topology. As mentioned before, the hexagonal-shaped processors still take a maximum of two inputs from the six nearest-neighbor processors. Compared with the design using 4-neighbor mesh, seven routing processors are saved, which accounts for a 22% processor number reduction.

Figure 8(a) shows the number of processors used for mapping the two applications to all three topologies. The 6-neighbor hex grid and 8-neighbor mesh are much more efficient than the baseline 2D mesh, resulting in a number of processor savings of 25% and 22% for the H.264 residual encoder and both 25% for the 802.11a receiver. The 8-neighbor mesh requires slightly larger number of processors than the 6-neighbor hex grid topology which yields the largest reduction (24%) in average number of used processors compared to 4-neighbor mesh. This is because the communication patterns

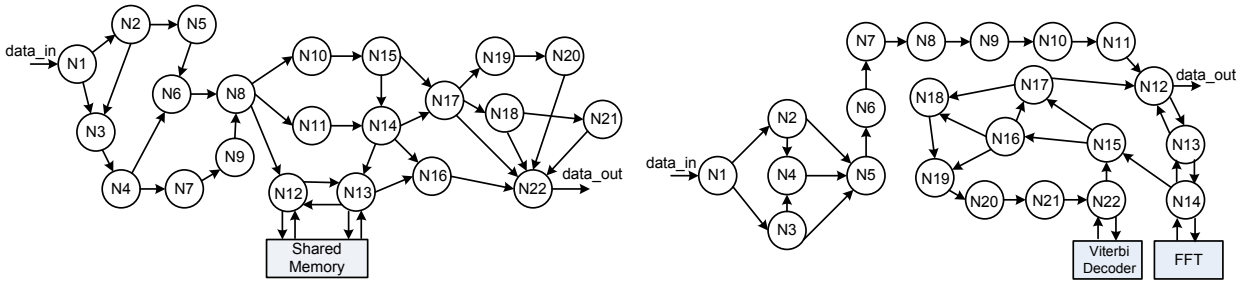

Fig. 5. Task graph of (a) a 22-node H.264/AVC video residual encoder, and (b) a 22-node 802.11a WLAN baseband receiver.

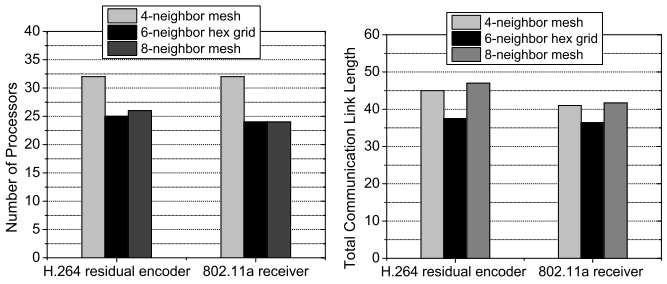

Fig. 8. The application mapping results of the 4-neighbor mesh, 6-neighbor hex grid and 8-neighbor mesh (a) the number of used processors, (b) the total communication link length.

of the two applications are mostly localized. Thus, topologies with more nearest-neighbor links yield more benefits than topologies with less nearest-neighbor links.

Figure 8(b) shows the total communication link length for the two applications which is calculated based on the data in Table I and the application mapping diagrams. The 8-neighbor mesh has longer communication length than the 4-neighbor mesh because of using more long communication links. The 6-neighbor hex grid is the most efficient topology, yielding the largest reduction (19%) in average total communication link length compared to the baseline 4-neighbor mesh.

V. Physical Design Methodology and Hexagonal Processor Tile Design

A. Physical Design Methodology

For performance evaluation, a small DSP processor with configurable circuit-switch interconnection is used for all physical designs. The processor contains a 16-bit datapath with a 40-bit accumulator and 560-Byte instruction and 256-Byte data memories. Each processor also contains two 128-Byte FIFOs for data buffering and synchronization between two processors. Each set of inter-processor links are composed of 19 signals including a clock, 16-bit data and 2 flow-control signals. This processor is tailored for all topologies under test with a different number of neighboring interconnections ranging from 4 to 8. The internal switch fabrics are changed accordingly. The hardware overhead is minimal for 6-neighbor and 8-neighbor processors with only 0.7% and 2.0% hardware overhead based on the synthesis results. In order to make CMP integration simpler, four additional sets of pins are inserted into the processor netlist after synthesis and are directly connected with bypass wires for the 8-neighbor processor. This adds routing congestion in the corner for the 8-neighbor mesh topology shown in Figure 2(c).

The processors are synthesized from Verilog with Synopsys Design Compiler and laid out with an automatic timing-driven physical design flow with Cadence SoC Encounter in 65 nm CMOS technology. Timing is checked and optimized after each step of the physical design flow: floorplan, power planning, cell placement, clock tree insertion and detailed routing.

B. Hexagonal Processor and CMP Design

The hexagonal-shaped tile bring challenges for physical implementation. The first challenge to design the hexagonal processor is how to create a hexagonal shape at the floorplan stage. The rectangular placement and routing blockage in SoC Encounter are used to create approximate triangle corner blockages with each rectangular blockage differs by one unit in width and height. All rect blockages are piled together to create an approximate triangle in the four corners of the rectangular floorplan as shown in Figure 9.

A proper placement of pins can help to achieve efficient global routing and easy CMP integration. At the floorplan stage, four sets of pins are put along the diagonal edge of the corner and two set of pins are placed in the horizontal top and bottom edge. Since all macroblocks have rectangular shapes (IMEM, DMEM and two FIFOs), this presents a challenge to place the macroblocks. In this design, the macroblocks are placed along the edge and the IMEM is placed in the right corner, respectively as shown in Figure 9.

The metal 6 and metal 7 are used to distribute power over the chip and the automatically-created power stripes can stop at the created triangle edge in the corner. The power pins are created on the top and bottom horizontal edges. When integrating the hexagonal processor together, the power nets along the triangle edge can be connected automatically or manually by simple abutment.

Once a hexagonal processor tile is laid out, a script is used to generate the RTL files of the multiprocessor. The CMP array can be synthesized with empty processor tiles inside. Another

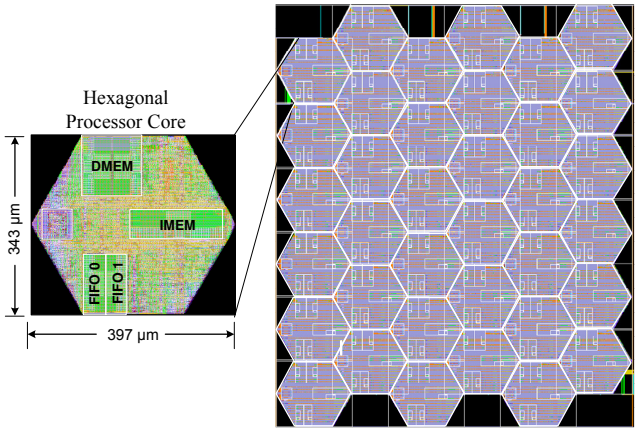

Fig. 9. Layout of a hexagonal processor and a 6x6 multiprocessor array.

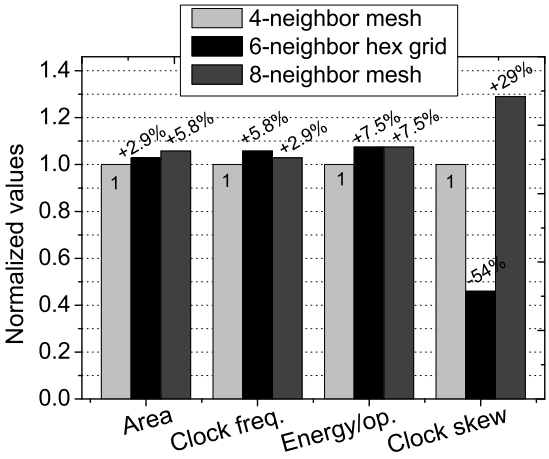

Fig. 10. Comparison of key metrics of the three optimized processor tile layout: normalized area, maximum clock frequency, energy per operation, and clock skew.

script places the hexagonal tiles with the blockage area overlap with nearest-neighbor processors along the triangle edge of each hexagonal tile. The SoC Encounter can connect all pins automatically although there are overlaps between LEF (library exchange format) files. The final GDSII files are read into Cadence icfb for design rule check (DRC). Figure 9 shows the final layout of a hexagonal-shaped processor tile and a 6 by 6 hexagonal-tiled multiprocessor array. There are small empty spaces along the edges of the chip as described in Section III.

VI. Experiment Results

A. Processor Implementation

All discussed topologies enable an easy integration of processors by abutment without global wires in the physical design phase. For all topologies, there is no long-distance inter-communication link across more than two processors and the processor has been pipelined in a way that the critical path is not in the interconnection links. Therefore, the maximum achievable frequency of an array is the same as an individual core, which is one of the key advantages of our proposed dense on-chip networks. Three tile types are implemented from RTL to GDSII layout. In order to be fair, all floorplans use the same power distribution design and the I/O pins and macroblocks are placed along edges reasonably depending on the topology.

In standard-cell design, the cell utilization ratio has a strong impact on the implementation result. A higher cell utilization can both save area and increase system performance if the design is routable. In order to get a minimum chip area for all tiles, we start with a relatively large tile area which results in a small cell utilization ratio. Then the tiles are repeatedly laid out while maintaining the aspect ratio and reducing the area by 5% in each iteration with minor pin and macroblock position adjustments in the floorplaning phase. Once a minimum area within 5% has been reached, the area change is reduced to 2.5%. The layout tool is pushed until it is not able to generate an error-free GDSII layout for all tiles. Our methodology results in a high cell utilization for all three tiles ranging from 81% to 83%.

Figure 10 shows the normalized implementation results of the three processor tiles in terms of area, max clock frequency, energy per operation and clock skew. The baseline 4-neighbor rectangular tile has the smallest area and the highest cell utilization of 83%. Compared with the baseline 4-neighbor rectangular tile, an area increase of 2.9% and 5.9% are required for the 6-neighbor hexagonal-shaped tile and the 8-neighbor rectangular tile, respectively. Both designs have a cell utilization of 81%.

Figure 10 also depicts the normalized maximum clock frequency relative to the baseline 4-neighbor rect tile which can operate at a maximum of 1065 MHz at 1.3 V. Due to an increase of area, the 8-neighbor rect tile can operate at 2.9% higher frequency than the 4-neighbor rect tile. The 6-neighbor hexagonal-shaped tile has noticeably higher frequencies than baseline 4-neighbor rect tile, which achieves a frequency increase of 5.8%.

Figure 10 shows the energy per operation for all tiles, which is estimated based on a 20% activity factor for all internal nodes. Both the 6-neighbor hex tile and 8-neighbor rect tile have a higher energy per operation (7.5%) because of the extra circuits for interconnections.

As for clock skew, the 8-neighbor rect tile shows a 29% higher clock skew probably because routing congestion in the corners affects the clock tree synthesis. The more circular-like shape helps the layout tool for a clock tree insertion and the hexagonal-shaped tile achieves the lowest clock skew with a reduction of 54% compared to the baseline 4-neighbor rect tile.

B. Application Area and Power

The actual application area depends on the number of used processors and the processor tile sizes. Figure 11 shows the normalized application area of the H.264 residual encoder and the 802.11a baseband receiver for all three topologies. The average application area reductions are 21% and 18% for the 6-neighbor hex grid topology and the 8-neighbor mesh topology, respectively. Corresponding to the largest reduction of the number of used processors, 6-neighbor hex grid topology achieves the largest application area reduction.

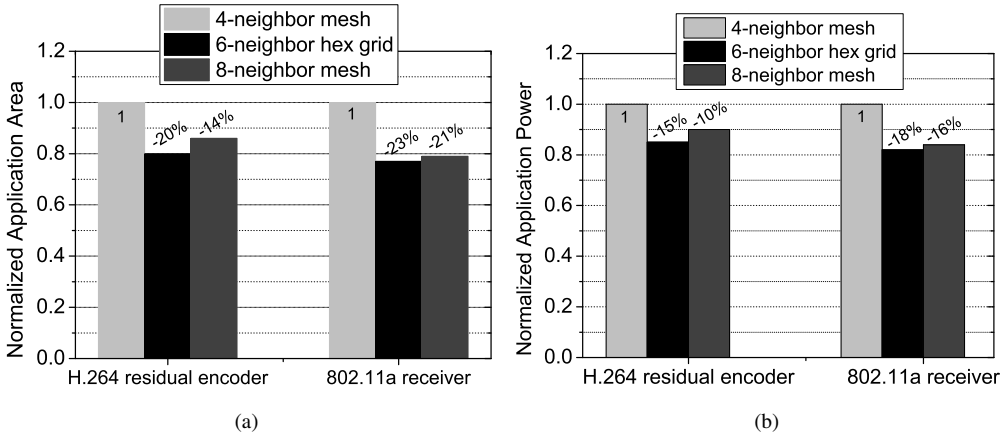

Fig. 11. The final mapping results of the H.264 residual encoder capable of HD 1080p encoding at 30 fps and 802.11a baseband receiver in 54 Mbps mode (a) normalized application area, and (b) normalized power consumption.

Since tightly-tiled architecture does not have global long wires, the total application power depends on the number of used processors and the computational workload for each processor tile. In order to meet the throughput requirement for the two mapped applications, processors need to run at 959 MHz at a supply voltage of 1.15 V for H.264 residual encoder and 594 MHz at a supply voltage of 0.92 V for 802.11a baseband receiver. Based on the processor power consumption numbers, application mapping diagrams and the required clock frequencies and supply voltages for processors, Figure 11(b) shows the normalized estimated average power consumption of the H.264 residual encoder (processing 1080p video at 30 fps) and the 802.11a baseband receiver (54 Mbps mode) for all three topologies. Compared to 4-neighbor mesh topology, the average application power reductions are 17% and 13% for the 6-neighbor hex grid and the 8-neighbor mesh topology, respectively. The 6-neighbor hex grid is the most power-efficient topology among all three topologies.

VII. Conclusion

This paper presents two low area overhead and low design complexity topologies other than the commonly-used 2D mesh for tiled many-core architecture. The proposed topologies include one 6-neighbor topology which uses novel hexagonal-shaped processor tiles. This work demonstrates the feasibility of using commonly available commercial CAD tools to implement CMPs with hexagonal processor tiles. Compared to 4-neighbor 2D mesh, the proposed 6-neighbor hex grid topology has little performance and energy penalties and small area overhead while providing much more effective inter-processor interconnect to reduce application area, power consumption and total communication link lengths.

References

[1] M. Horowitz R. Ho, K. Mai, "The future of wires," *Proc. of IEEE*, vol. 89, pp. 490–504, Apr. 2001.

[2] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Effect of traffic localization on energy dissipation in NoC-based interconnect," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, 2005, pp. 1774–1777.

[3] M. Taylor et al., "A 16-issue multiple-program-counter microprocessor with point-to-point scalar operand network," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2003, pp. 170–171.

[4] Zhiyi Yu, M.J. Meeuwsen, R.W. Apperson, O. Sattari, M. Lai, J.W. Webb, E.W. Work, D. Truong, T. Mohsenin, and B.M. Baas, "AsAP: An asynchronous array of simple processors," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 3, pp. 695–705, Mar. 2008.

[5] S. Bell et al., "TILE64 processor: A 64-core soc with mesh interconnect," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2008, pp. 88–89.

[6] Dean N. Truong, Wayne H. Cheng, Tinoosh Mohsenin, Zhiyi Yu, Anthony T. Jacobson, Gouri Landge, Michael J. Meeuwsen, Anh T. Tran, Zhibin Xiao, Eric W. Work, Jeremy W. Webb, Paul V. Mejia, and Bevan M. Baas, "A 167-processor computational platform in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1130–1144, Apr. 2009.

[7] J. Howard, S. Dighe, S.R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V.K. De, and R. Van Der Wijngaart, "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 173 –183, Jan. 2011.

[8] Allen D. Malony, "Regular processor arrays," in *the 2nd Symposium on the Frontiers of Massively Parallel Computation*, 1988, pp. 499–502.

[9] M.-S. Chen, K.G. Shin, and D.D. Kandlur, "Addressing, routing, and broadcasting in hexagonal mesh multiprocessors," *IEEE Transactions on Computers*, vol. 39, pp. 10–18, 1990.

[10] Catherine Decayeux and Davis Seme, "3D hexagonal network: modeling, topological properties, addressing scheme, and optimal routing algorithm," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 9, pp. 875–884, Sep. 2005.

[11] Ivan Stojmenovic, "Honeycomb networks: Topological properties and communication algorithms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, pp. 1036–1042, 1997.

[12] A.W. Yin, T.C. Xu, P. Liljeberg, and H. Tenhunen, "Explorations of honeycomb topologies for network-on-chip," in *Network and Parallel Computing, 2009. NPC '09. Sixth IFIP International Conference on*, Oct. 2009, pp. 73–79.

[13] Zhiyi Yu and B.M. Baas, "A low-area multi-link interconnect architecture for GALS chip multiprocessors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 5, pp. 750–762, may. 2010.

[14] Zhibin Xiao and Bevan Baas, "A 1080p H.264/AVC baseline residual encoder for a fine-grained many-core system," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 21, no. 7, pp. 890–902, 2011.

[15] Anh T. Tran, Dean N. Truong, and Bevan M. Baas, "A complete real-time 802.11a baseband receiver implemented on an array of programmable processors," in *Asilomar Conference on Signals, Systems and Computers (ACSSC)*, Oct. 2008, pp. 165–170.