A Hexagonal Processor and Interconnect Topology for Many-Core Architecture with Dense On-Chip Networks

Zhibin Xiao and Bevan Baas

Department of Electrical and Computer Engineering
University of California, Davis
1 Shields Avenue Davis, CA USA 95616
{zxiao,bbaas}@ucdavis.edu

Abstract. Network-on-Chips (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. Motivated by the fact that applications often have largely localized communication patterns, we have proposed an 8-neighbor mesh topology and a 6-neighbor topology with hexagonal-shaped processor tiles, both of which increase local connectivity while keep much of the simplicity of a mesh-based topology. We have physically designed a 16-bit DSP processor and the corresponding processor arrays which utilize all three topologies. 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%.

Keywords: CMOS, many-core processor, interconnection topology, network on chip (NoC), digital signal processing (DSP).

1 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

A. Burg et al. (Eds.): VLSI-SoC 2012, IFIP AICT 418, pp. 125–143, 2013.

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 as shown in Figure 1(a). 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.

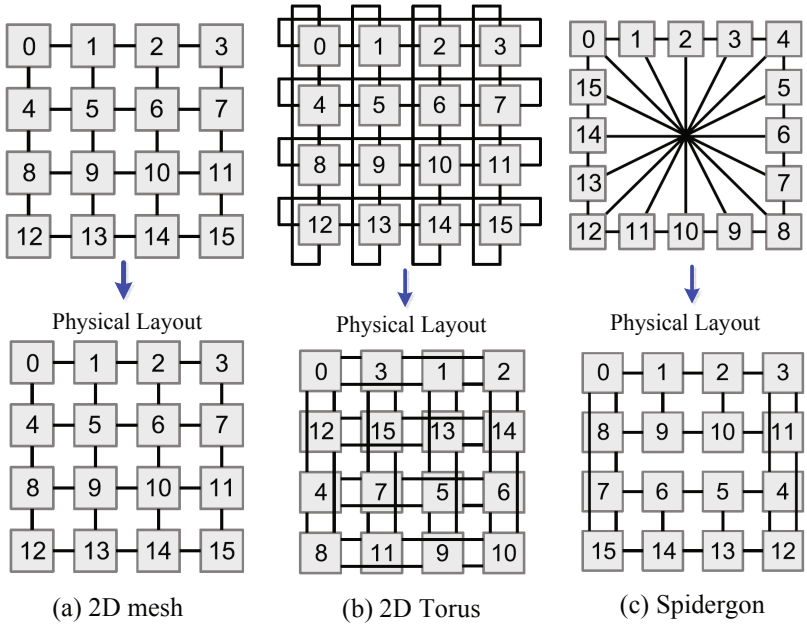


Fig. 1. Popular Network-on-Chip topologies and their physical layouts [2]: (a) 2D mesh, (b) 2D Torus, and (c) Spidergon

There exist other common topologies for NoCs such as 2D torus, Spidergon, fat tree and higher dimensional meshes and tori which provide higher routing capability and communication bandwidth with costs of higher wire density and longer global wires. Furthermore, topologies with irregular layouts present significant challenges for many-core implementations especially with the number of

cores per die expected to soon reach thousands and more. As an example, Figure 1(b)(c) shows the 2D torus and Spidergon topologies as well as their physical layouts on a 2-dimensional chip [2]. Both topologies require global wires which go across one or more processors. Mapping arbitrary non-regular topologies to a 2D floorplan is an NP-hard optimization problem [3].

For many applications mapped onto homogeneous chip multiprocessors, communication within processors is often largely localized [4], 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 2 describes the related work. Section 3 presents the proposed inter-processor communication topologies. Section 4 shows the mapping of two complex applications to all discussed topologies. In section 5, the physical design of the hexagonal-shaped processor tiles is presented. Section 6 presents the chip implementation results and section 7 concludes this paper.

2 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 [5], AsAP [6], TILE64 [7], AsAP2 [8] and Intel 48-core Single-Chip Cloud Computer (SCC) [9].

Becker et al. [11] developed a hexagonal Field-programmable Analog Array in a 0.13 μm CMOS technology. The basic building block is a hexagonal analog circuit block which communicates with six neighbors. Extension to a many-core processor is similar in topology, but very different in terms of impact on tile area and total application interconnect.

Malony studies the two-dimensional regular processor arrays which are geometrically defined based on nearest-neighbor connections and space-filling properties [12]. 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

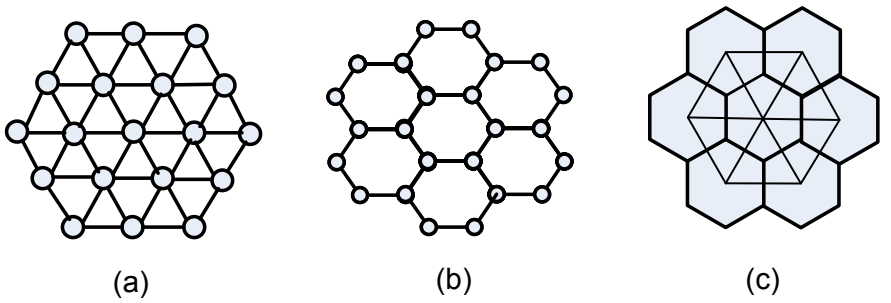


Fig. 2. Examples of three hexagonal networks (a) a 6-neighbor off-chip hexagonal network; (b) a 3-neighbor on-chip honeycomb network [10]; (c) the proposed 6-neighbor on-chip hexagonal grid network

in hexagonal mesh multiprocessors [13]. Decayeux and Seme proposed a 3D hexagonal network as an extension of 2D hexagonal networks [14]. 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 2(a). Stojmenovic proposed efficient coordinate system and routing algorithms for the 3-neighbor honeycomb mesh networks as shown in Figure 2(b) [15]. 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 2(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.

3 Processor Shapes and Topologies

3.1 NoC Topology Analysis Criteria

NoC topologies can be analyzed by a few criteria [16]:

- *Degree*: is the number of direct neighbors for one node. A high degree allows more nodes to communicate directly with low latency.
- *Diameter*: is the largest number of hops between any two nodes. A small diameter indicates low maximum latency of a network.
- *Bisection*: is the minimum number of links to be removed to separate a network into two equal ones. A high bisection indicates a high bandwidth yielding high throughput.
- *Number of links*: the total number of bidirectional links in a network.
- *Clustering degree*: also called clustering coefficient, is a measure of degree to which nodes in a network tend to cluster together. The local clustering degree for a node i can be defined as: $\frac{2l_i}{n_i(n_i-1)}$, where n_i is the number of direct neighbors and l_i is the number of links between its neighboring nodes. A high clustering degree indicates that local nodes close to each other are strongly connected.

Table 1. Characteristics of various regular topologies for a homogenous many-core array with $n \times n$ processors where n is the number of processors on one edge and $n \geq 2$

Topology	Degree	Max. Link Hops	Link Num.	Diameter	Bisection	Clustering Degree
2D Mesh	4	0	$2n(n-1)$	$2(n-1)$	n	0
2D Torus	4	1	$2n^2$	n	$2n$	0
8-neighbor mesh	8	1	$4n^2 - 6n + 2$	$n-2$	$3n-2$	0.86
6-neighbor hexagon	6	0	$3n^2 - 4n + 1$	$n + \lfloor \frac{n-2}{2} \rfloor$	$2n-1$	0.40

[+] Omitted due to space limitation. The total number of links for 5-5 House and Rect is: $n(n-1) + n(\lfloor \frac{n-1}{2} \rfloor) + (2n-1)(\lceil \frac{n-1}{2} \rceil)$.

[*] This is for $n \geq 4$. If $n \leq 3$, the diameter of the topology is: $n + \lfloor \frac{n-1}{2} \rfloor$.

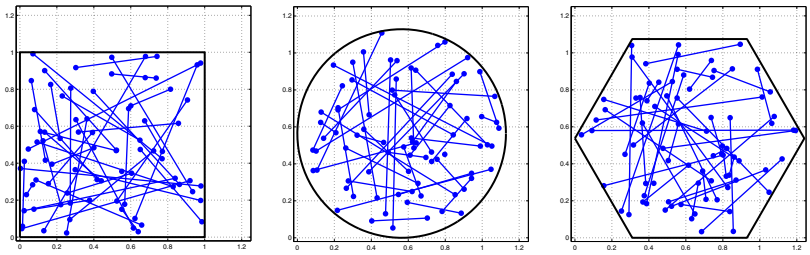
– *Max link hops*: is the maximum hops that a link can cross after the topology has been physically mapped to a 2-dimensional chip. This is a criteria proposed in this work to measure the length of global wires of a topology.

The above criteria can be used to compare various topologies and provide an initial indication on performance. The first two rows of Table 1 list the characteristics of two popular topologies 2D mesh and 2D torus. 2D mesh has a maximum degree of 4 and a maximum link hop equal to 0 since all of the links are nearest-neighbor. For an $n \times n$ array, 2D mesh has a number of links equal to $2n(n-1)$, a diameter equal to $2(n-1)$, bisection n , and a clustering degree equal to 0. Compared with 2D mesh, 2D torus has the same degree, more links, smaller diameter, higher bisection bandwidth and the same clustering degree. All of these criteria indicate 2D torus could achieve higher throughput and lower latency at the cost of more long non-nearest neighbor links.

This work explores low complexity topologies with higher degree, larger number of links, smaller diameter, higher bisection compared to 2D mesh. We also limit the maximum link hops being less than or equal to one to avoid global long wires. These requirements result in proposed topologies that have a strong local connectivity with a non-zero clustering degree. In the following subsections, a 6-neighbor hexagon and an 8-neighbor mesh topologies are proposed and analyzed.

3.2 Processor Tile Shapes

To the best of our knowledge, all previously-fabricated VLSI processors have been of a rectangular shape, often nearly square. As illustrated in Figure 3(a)(b), it stands to reason that a circular shape would allow shorter wires for a given netlist, resulting in smaller area and lower wire capacitance which would result



(a) Square (b) Circle (c) Hexagon

Fig. 3. Example tiles of constant area with random uniformly-distributed wire endpoints

in higher speeds and lower energy per operation. A simple experiment with ideal shapes and one million randomly-placed wires yields a 2.2% reduction in total wire length for a circular tile compared to a square tile. On the negative side, it is clear that circles do not pack together without wasted space between tiles. On the positive side, circles pack with *six* neighbors while rectangles obviously have only four. It is reasonable to expect a rectangular tile to have longer wires on average compared to a square tile.

In contrast to the circle, the hexagonal shape *does* pack efficiently without gaps between tiles and it retains the 6-nearest-neighbor property. The same wiring experiment was run for a hexagonal tile and it resulted in a 1.8% reduction in total wire length compared to the square tile. A reduction in total wire length yields a pure benefit in area, energy and delay for processor tile design. The inclusion of common rectangular blocks such as memory arrays in a processor tile increases routing congestion but is shown in Section 6 to be tolerable. In addition, we demonstrate that Manhattan-style wire routing is fully compatible with non-rectangular tile shapes.

3.3 The Proposed Topologies

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

Figure 4(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

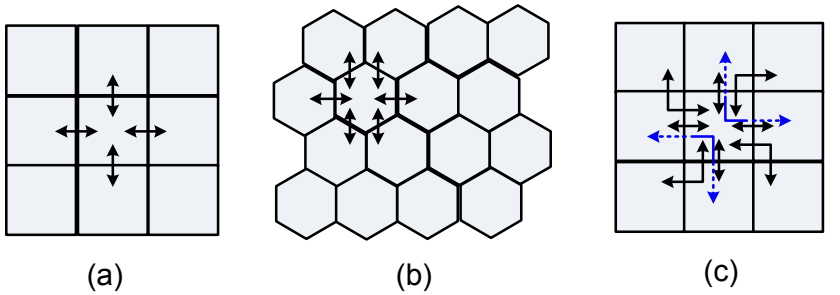


Fig. 4. The three inter-processor communication topologies considered in this work: (a) baseline 4-neighbor mesh (b) 6-neighbor hexagonal tile and interconnect, and (c) 8-neighbor mesh

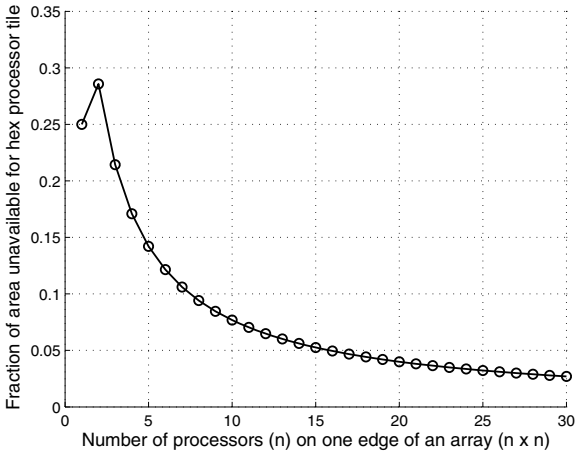


Fig. 5. Fraction of area unavailable for processor tiles in an n x n many-core array utilizing 6-neighbor hexagonal tiles and interconnect topology

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. Figure 5 shows the percentage of unavailable area for the hexagonal-shaped tile topology with varying processor array size. If the 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 4(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 4(c)).

Table 2. 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

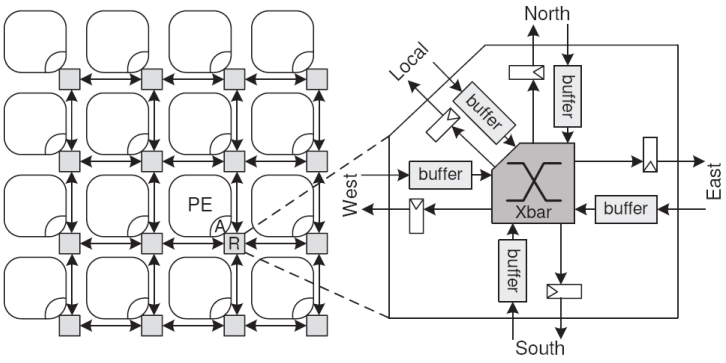


Fig. 6. A 2D mesh processor array connected by a dynamic five-port routers each with one port connected to the processor core

Table 1 also lists the characteristics of the two proposed topologies for an homogenous many-core array with $n \times n$ processors. Compared with 2D mesh, the two proposed topologies have larger node degree, smaller diameter, larger or equal bisection bandwidth and larger clustering degree. The 8-neighbor mesh topology has the largest bisection, smallest diameter and largest clustering degree, which indicates lower maximum latency and high maximum throughput. The advantage of the 6-neighbor hexagon topologies is that global long wires are not required.

The center-to-center distance can be used to represent the communication link length between two "touched" processors. Table 2 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 2, 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.

4 Application Mapping

4.1 Target Interconnect Architecture

Fig. 6 shows the inter-processor communication in a typical 2D mesh processor array using dynamic routers. As the diagram shows, the processor array is

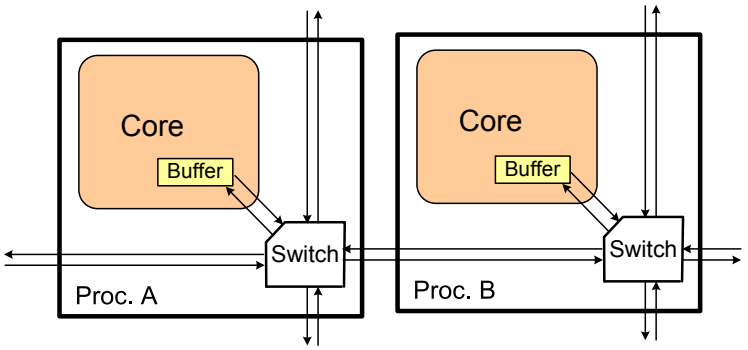


Fig. 7. A 2D mesh processor array connected by circuit switches each with four nearest-neighbor inter-processor communication links and one port connected to the processor core

connected by 5-port routers and the communication logic includes five buffers and one 5×5 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 interconnection while trading off routing flexibility [17].

Fig. 7 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 4×1 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. 7 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.

4.2 Application Mapping Methodology

Parallel programming on the discussed many-core systems with dense on-chip networks includes two main 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 [18]. The two steps might be repeated iteratively for throughput optimization where we can identify the bottleneck task of the design and partition it even more until the throughput meets the requirement.

To be specific, in the partitioning step, an estimate of task workload and required resources such as data and instruction memories are used to generate a fine-grained task graph where each task can be assigned to one processor node. Following the fine-grained partition, the mapping is conducted either manually or

automatically by an automatic mapping tool [19]. Application mapping is essentially an optimization problem, which can be formed as integer linear programming (ILP) problem [20] and solved by Heuristic algorithms such as simulated annealing [21]. In this work, we have used a manual mapping method and the primary optimization target is to minimize area and maximize local communication.

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. In order to be fair to compare all topologies, we chose not to partition tasks specifically for one topology and mapping the two applications onto all topologies is based on the same task graph.

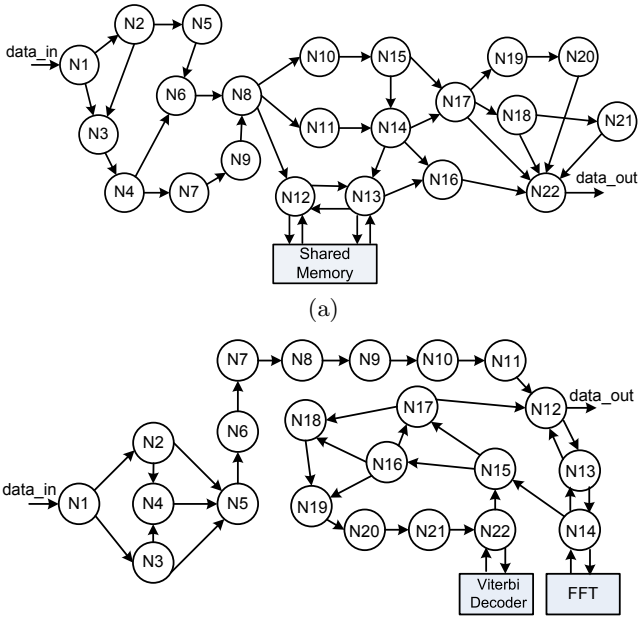


Fig. 8. Task graph of (a) a 22-node H.264/AVC video residual encoder, and (b) a 22-node 802.11a WLAN baseband receiver

4.3 Benchmark Application Mapping

Figure 8 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 8(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 [18]. The H.264/AVC encoder is a memory-intensive application which requires an additional shared memory module as shown in

the task graph. Figure 8(b) shows a 22-node task graph of a complete 802.11a WLAN baseband receiver which is computation-intensive requiring two dedicated hardware 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 [22]. Figure 9 shows an example mapping of the

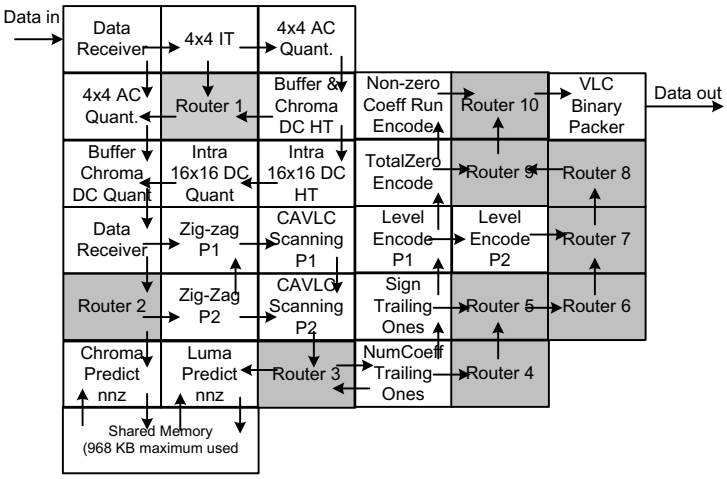


Fig. 9. An H.264/AVC video residual encoder mapped on a processor array with 4-neighbor 2D mesh topology

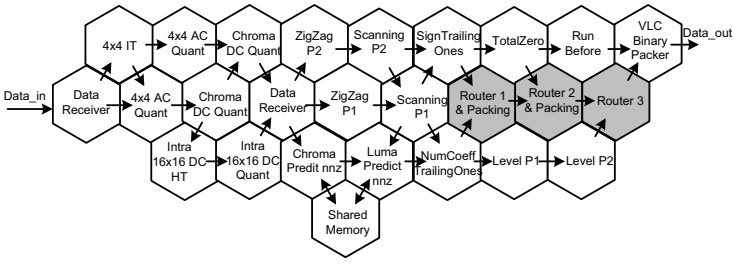


Fig. 10. An H.264/AVC residual video encoder mapped on a processor array with 6-neighbor hex topology

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 10 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

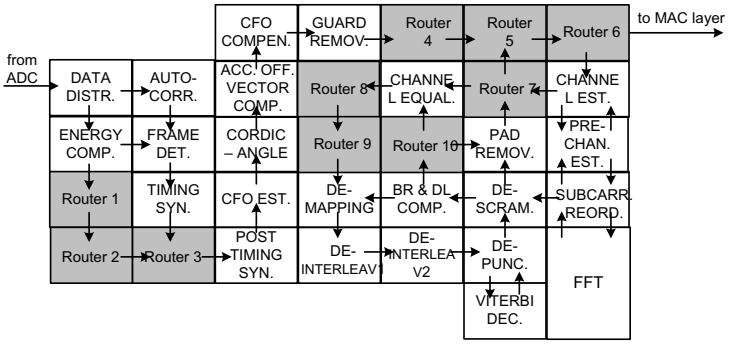


Fig. 11. An 802.11a baseband receiver mapped on the processor array with baseline 4-neighbor 2D mesh topology

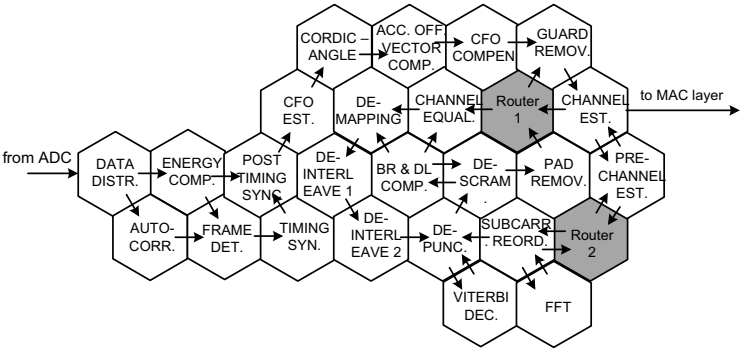


Fig. 12. An 802.11a baseband receiver mapped on the processor array with 6-neighbor hex topology

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.

Fig. 11 shows a mapping of the 802.11a/g baseband receiver (54 Mbps) on the baseline 4-neighbor 2D mesh that uses 32 processors plus the Viterbi decoder and FFT accelerators with 10 processors used for merging and forwarding data. Fig. 12 shows a mapping on the hexagonal-shaped tile architecture which requires only 24 processors plus the Viterbi decoder and FFT accelerators—25% fewer processors than those used in the 2D mesh mapping.

4.4 Application Mapping Results

Figure 13(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 of the two applications are mostly localized. Thus, topologies with more nearest-neighbor links yield more benefits than topologies with less nearest-neighbor links.

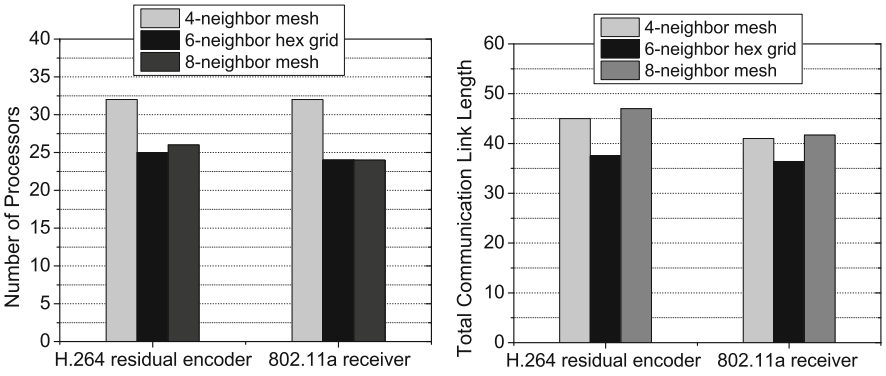


Fig. 13. The application mapping results of the 4-neighbor mesh, 6-neighbor hex grid and 8-neighbor mesh (a) the number of used processors, (b) the total communication link length

Figure 13(b) shows the total communication link length for the two applications which is calculated based on the data in Table 2 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.

5 Physical Design Methodology and Hexagonal Processor Tile Design

5.1 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 Fig. 4(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.

5.2 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 Fig. 14.

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 Fig. 14.

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 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). Fig. 14 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 3.

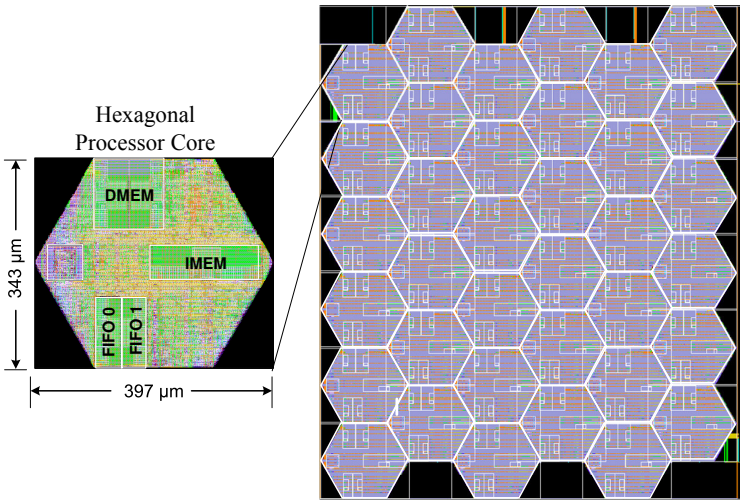


Fig. 14. Layout of a hexagonal processor and a 6x6 multiprocessor array

6 Experiment Results

6.1 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 15 shows the normalized implementation results of the three processor tiles in terms of area, max clock frequency, energy per operation and clock skew.

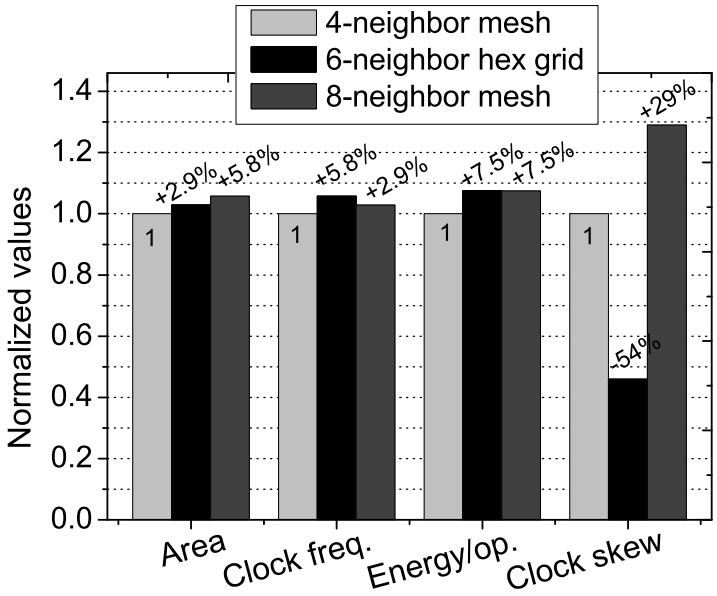


Fig. 15. Comparison of key metrics of the three optimized processor tile layout: normalized area, maximum clock frequency, energy per operation, and clock skew

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 15 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 15 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.

6.2 Application Area and Power

The actual application area depends on the number of used processors and the processor tile sizes. Fig. 16 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.

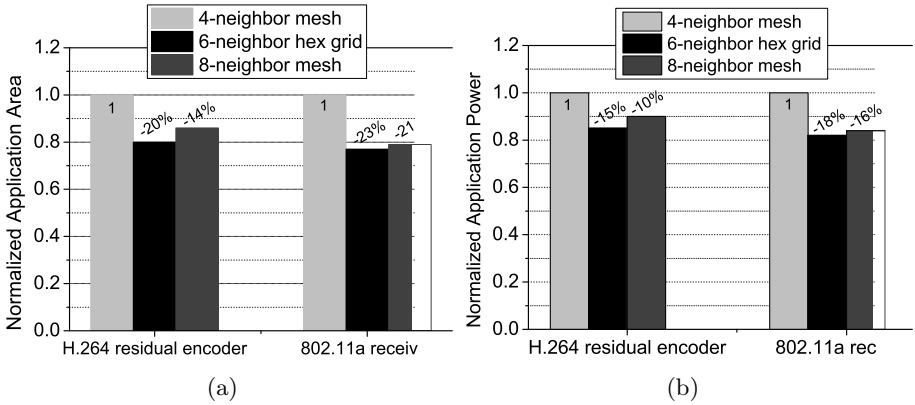


Fig. 16. The final mapping results of the H.264 residual encoder capable of HD 1080p encoding at 30 fps and 802.11a baseband receiver in 54 Mbps mode (a) normalized application area, and (b) normalized power consumption

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, Fig. 16(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.

7 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.

Acknowledgments. The authors gratefully acknowledge support from ST Microelectronics, Intel, UC Micro, NSF Grant 0430090 and CAREER Award 0546907, SRC GRC Grant 1598, CSR Grant 1659, Intellasys, S Machines and the support of the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity. The authors thank Dean Truong for the chip layout assistance and Anh Tran for providing the 2D mesh mapping of the 802.11a WLAN baseband receiver, P. Cogez, and E. Flamand.

References

1. Ho, R., Mai, K., Horowitz, M.: The future of wires. Proc. of IEEE 89, 490–504 (2001)
2. Neeb, C., Wehn, N.: Designing efficient irregular networks for heterogeneous systems-on-chip. In: 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD 2006), pp. 665–672 (2006)
3. Leary, G., Srinivasan, K., Mehta, K., Chatha, K.: Design of network-on-chip architectures with a genetic algorithm-based technique. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 17(5), 674–687 (2009)
4. Pande, P.P., Grecu, C., Jones, M., Ivanov, A., Saleh, R.: Effect of traffic localization on energy dissipation in NoC-based interconnect. In: Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), pp. 1774–1777 (2005)
5. Taylor, M., et al.: A 16-issue multiple-program-counter microprocessor with point-to-point scalar operand network. In: IEEE International Solid-State Circuits Conference (ISSCC), pp. 170–171 (February 2003)
6. Yu, Z., Meeuwsen, M., Apperson, R., Sattari, O., Lai, M., Webb, J., Work, E., Truong, D., Mohsenin, T., Baas, B.: AsAP: An asynchronous array of simple processors. IEEE Journal of Solid-State Circuits 43(3), 695–705 (2008)
7. Bell, S., et al.: TILE64 processor: A 64-core soc with mesh interconnect. In: IEEE International Solid-State Circuits Conference (ISSCC), pp. 88–89 (February 2008)
8. Truong, D.N., Cheng, W.H., Mohsenin, T., Yu, Z., Jacobson, A.T., Landge, G., Meeuwsen, M.J., Tran, A.T., Xiao, Z., Work, E.W., Webb, J.W., Mejia, P.V., Baas, B.M.: A 167-processor computational platform in 65 nm CMOS. IEEE Journal of Solid-State Circuits 44(4), 1130–1144 (2009)

9. Howard, J., Dighe, S., Vangal, S., Ruhl, G., Borkar, N., Jain, S., Erraguntla, V., Konow, M., Riepen, M., Gries, M., Droege, G., Lund-Larsen, T., Steibl, S., Borkar, S., De, V., Van Der Wijngaart, R.: 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 46(1), 173–183 (2011)
10. Yin, A., Xu, T., Liljeberg, P., Tenhunen, H.: Explorations of honeycomb topologies for network-on-chip. In: Sixth IFIP International Conference on Network and Parallel Computing, NPC 2009, pp. 73–79 (October 2009)
11. Becker, J., Henrici, F., Trendelenburg, S., Ortmanns, M., Manoli, Y.: A continuous-time hexagonal field-programmable analog array in 0.13um CMOS with 186MHz GBW. In: IEEE International Solid-State Circuits Conference (ISSCC), pp. 70–71 (February 2008)
12. Malony, A.D.: Regular processor arrays. In: The 2nd Symposium on the Frontiers of Massively Parallel Computation, pp. 499–502 (1988)
13. Chen, M.S., Shin, K., Kandlur, D.: Addressing, routing, and broadcasting in hexagonal mesh multiprocessors. IEEE Transactions on Computers 39, 10–18 (1990)
14. Decayeux, C., Seme, D.: 3D hexagonal network: modeling, topological properties, addressing scheme, and optimal routing algorithm. IEEE Trans. on Parallel and Distributed Systems 16(9), 875–884 (2005)
15. Stojmenovic, I.: Honeycomb networks: Topological properties and communication algorithms. IEEE Transactions on Parallel and Distributed Systems 8, 1036–1042 (1997)
16. Chariete, A., Bakhouya, M., Gaber, J., Wack, M.: An approach for customizing on-chip interconnect architectures in soc design. In: 2012 International Conference on High Performance Computing and Simulation (HPCS), pp. 288–294 (July 2012)
17. Yu, Z., Baas, B.: A low-area multi-link interconnect architecture for GALS chip multiprocessors. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 18(5), 750–762 (2010)
18. Xiao, Z., Baas, B.: A 1080p H.264/AVC baseline residual encoder for a fine-grained many-core system. IEEE Transaction on Circuits and Systems for Video Technology 21(7), 890–902 (2011)
19. Work, E.W.: Algorithms and software tools for mapping arbitrarily connected tasks onto an asynchronous array of simple processors. Master's thesis, University of California, Davis, CA, USA (September 2007), `http://www.ece.ucdavis.edu/vcl/pubs/theses/2007-4`
20. Tosun, S., Ozturk, O., Ozen, M.: An ILP formulation for application mapping onto network-on-chips. In: International Conference on Application of Information and Communication Technologies (AICT 2009), pp. 1–5 (October 2009)
21. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
22. Tran, A.T., Truong, D.N., Baas, B.M.: A complete real-time 802.11a baseband receiver implemented on an array of programmable processors. In: Asilomar Conference on Signals, Systems and Computers (ACSSC), pp. 165–170 (October 2008)