SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

June 2018, Vol. 61 060415:1–060415:11
https://doi.org/10.1007/s11432-017-9351-4

Special Focus on Flexible Electronics Technology

An intelligent partitioning approach of the system-on-chip for flexible and stretchable systems

Changqing XU*, Yi LIU & Yintang YANG

School of Microelectronics, Xidian University, Xi'an 710071*, China*

Received 27 October 2017/Revised 1 January 2018/Accepted 11 January 2018/Published online 19 April 2018

Abstract In this paper, we propose an intelligent partitioning approach of the system-on-chip (SoC) to improve the bendability and stretchability of flexible and stretchable systems. The proposed approach partitions the SoC intelligently into clusters of functional modules according to the communication flows and area constraint. Based on the communication volume between clusters, a heuristic algorithm is applied to map these clusters onto the 2D mesh network-on-chip (NoC) for co-optimization of communication energy and delay. Experimental results show that our approach can effectively partition the SoC into small ICs of the same size. The approach also reduces power consumption and communication delay by 10.64%–56.63% and 15.06%–50.30%, respectively.

Keywords functional module clustering, cluster mapping, SoC, flexible and stretchable systems

Citation Xu C Q, Liu Y, Yang Y T. An intelligent partitioning approach of the system-on-chip for flexible and stretchable systems. Sci China Inf Sci, 2018, 61(6): 060415, https://doi.org/10.1007/s11432-017-9351-4

1 Introduction

Flexible electronics have the potential to transform computing by enabling bendable and stretchable systems with arbitrary shapes [1]. Physical flexibility, combined with cost and weight advantages, opens a wide range of form factors and application areas, including wearable electronics, prosthetics, medical sensing, rollable displays, and the internet of things (IoT) [2]. Over the last decade, tremendous advances in the system-on-chip (SoC) design have enabled integration of a complete system, including a large variety of processing elements and memory, on a single die [3], thus making the design of arbitrarily shaped, flexible and stretchable systems with integrated sensing, computing and communication capabilities possible, as shown in Figure 1. The implementation of these systems is similar to that of SoC, but flexible circuit and off-the-shelf rigid integrated circuits (ICs) are integrated on the flexible and stretchable substrates and the wire are stretchable. Most of the sensors, displays and antennas based on current technology are expected to be implemented using purely flexible electronics [4]. Because the SoC, which includes an application processor, memory, and other ICs, is too rigid to bent and stretch, the island-bridge structure is proposed [5–8]. In this structure, a segmented mesh with active device islands is connected electrically or mechanically by thin polymer bridges with or without metal-interconnect lines, respectively, and the flexible substrates, such as polydimethylsiloxane (PDMS), are used to improve the bendability and stretchability of the resulting systems. These designs introduce materials and mechanical design strategies for classes of electronic circuits that offer extremely high bendability and stretchability.

* Corresponding author (email: changqingxu@stu.xidian.edu.cn)

© Science China Press and Springer-Verlag GmbH Germany 2018

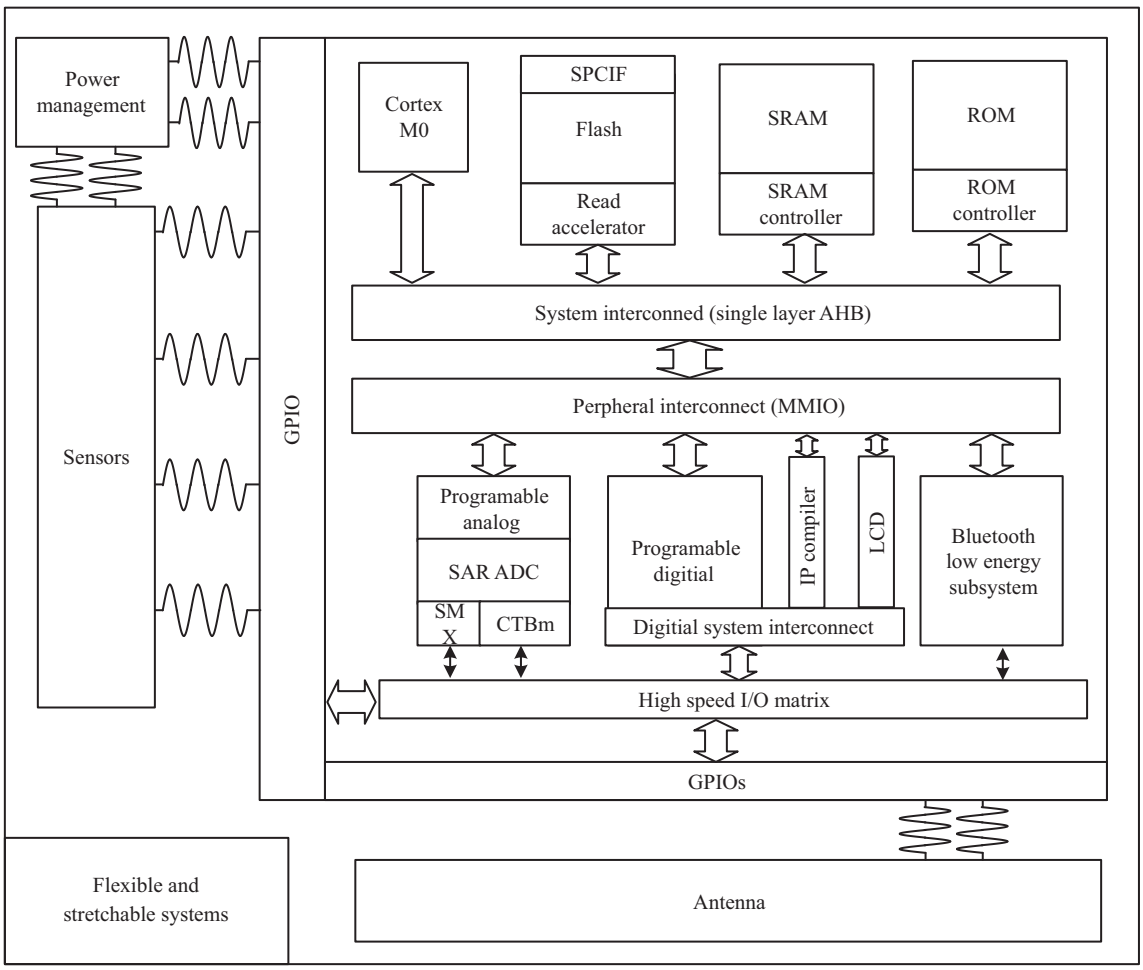

Figure 1 Illustration of flexible and stretchable systems with integrated sensing, computing and communication capabilities.

To illustrate the mechanical characteristics of the structure shown in Figure 1, the structure is modeled and analyzed in ANSYS 18.0. Figure 2 shows the system when it is bent and stretched. These two figures clearly show that rigid ICs degrade the bendability and stretchability of the flexible and stretchable system.

An effective approach to solve the problem of rigid ICs is to partition large SoC into small ICs. However, flexible and stretchable substrates features a limited number of layers, thus putting larger constraints on the number of wires and wire pitch available for partitioning. Besides manufacturing challenges, adding more wiring layers is not desirable because doing so decreases the flexibility of the system [3]. Flexibility is reduced as the number of wiring layers increases since the elasticity of interconnections is significantly smaller than that of the substrate. In work around these constraints, an approach based on the combination of network-on-chip (NoC) and stretchable interconnections was proposed in [3]. Figure 3 illustrates flexible and stretchable systems based on NoC. In the figure, a large rigid SoC is partitioned into nine small ICs, which are called tiles in this paper. A tile includes a router, links, and a network interface (NI). Routers transfer data over links between ICs, which are connected through an NI to the NoC. To illustrate the mechanical characteristics of the structure shown in Figure 3, the structure is modeled and analyzed in ANSYS 18.0. Figure 4 shows the system when it is bent and stretched. These two figures demonstrate that the bendability and stretchability of flexible and stretchable systems is improved when the size of the ICs is scaled down. The mesh topology also balances the force on interconnections when the system is bent or stretched. In [3], a SoC is partitioned into small ICs based on functional modules, however, the areas of the partitioned ICs different significantly, potentially leading to uneven forces on

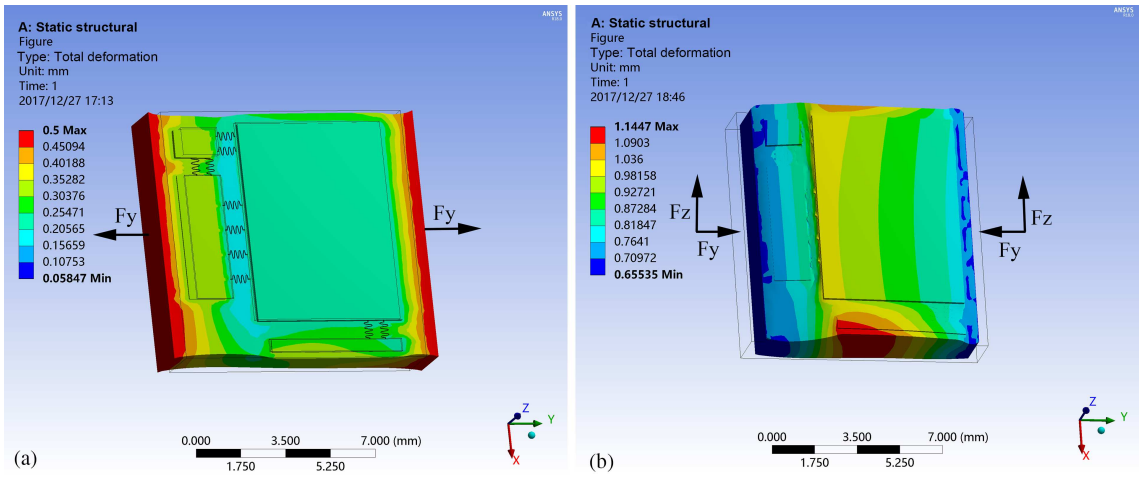

Figure 2 (Color online) Examples of bendable or stretchable systems. (a) Forces are applied in the y direction only; (b) forces are applied in the y and z directions only.

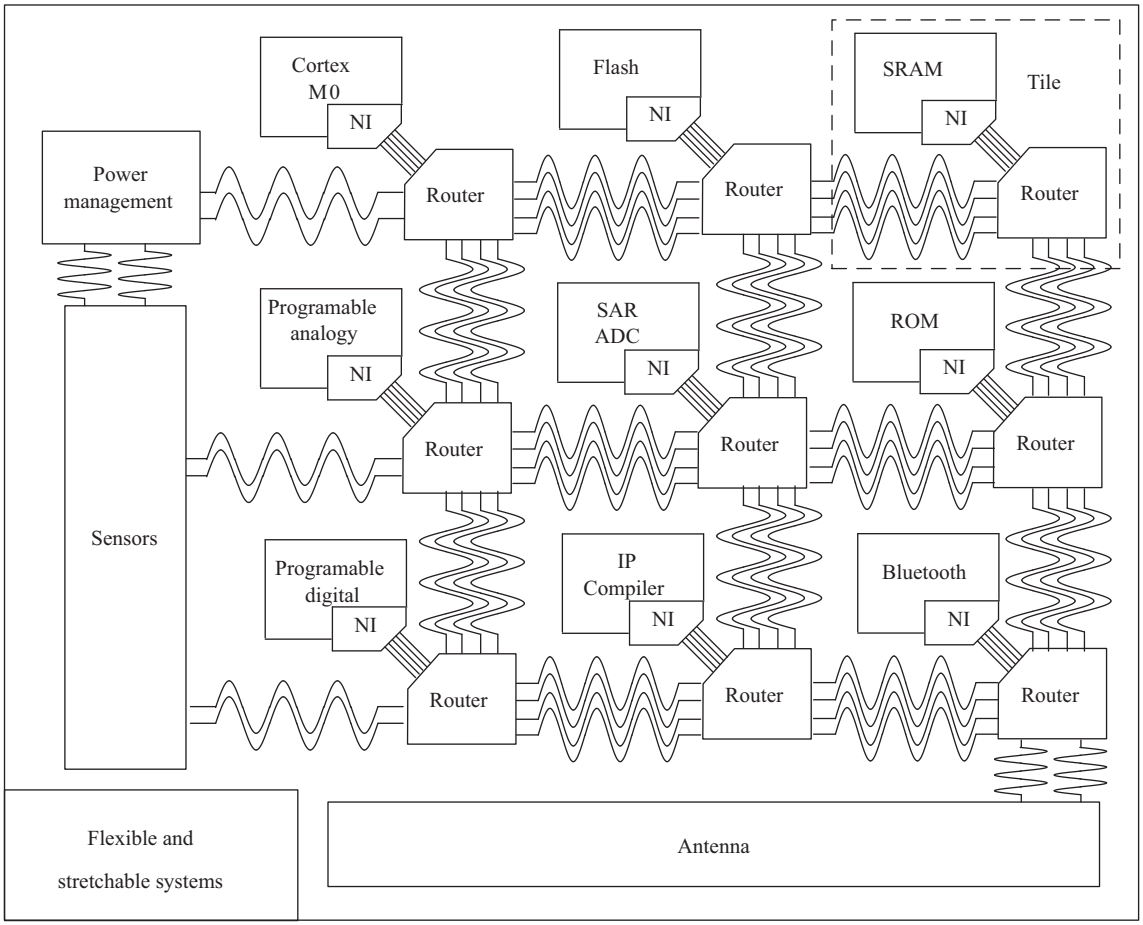

Figure 3 Illustration of flexible and stretchable systems based on network-on-chip.

interconnections that can break or permanently deform the contacts to the chip pins.

This paper presents a clustering based heuristic technology with which to partition the SoC in flexible and stretchable systems. The proposed approach partitions the SoC intelligently into clusters of functional modules according to the communication flows and area constraint. Based on the communication volume between the clusters, the resulting clusters are mapped onto the 2D mesh NoC. Because the problem of mapping is known to be NP-hard [9], a heuristic mapping algorithm called genetic based multi-operator

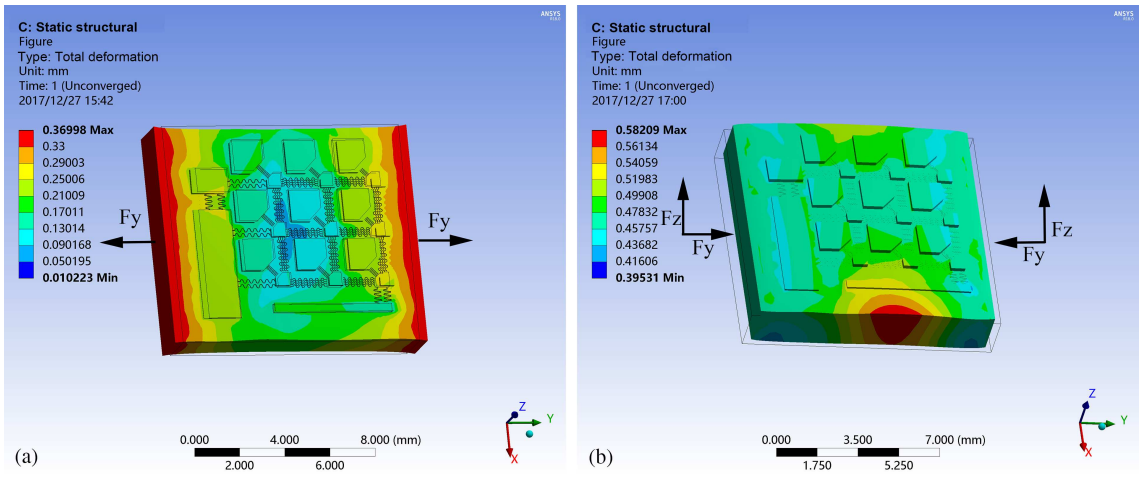

Figure 4 (Color online) Examples of bendable or stretchable systems. (a) Forces are applied in the y direction only; (b) forces are applied in the y and z direction only.

algorithm (GMO) is proposed for co-optimization of communication energy and delay. Compared with the approach in [3], our proposed approach can balance the area of partitioned ICs and effectively reduce communication energy and delay.

The rest of the paper is organized as follows. Section 2 presents an overview of the proposed partitioning approach. Section 3 provides the problem formulation and definitions. Section 4 introduces the partitioning algorithm. Section 5 shows the experimental results. Finally, Section 6 concludes this paper.

2 Partitioning approach

An overview of the proposed partitioning approach of the SoC for flexible and stretchable system is shown in Figure 5. This approach mainly includes two phases. The first phase is functional module clustering. The SoC can be naturally partitioned into functional modules. As the areas of these functional modules are very different, uneven forces may develop on interconnections. In this paper, the functional modules are divided into clusters based on their communication requirements and areas in the first phase. Because the number of clusters exerts a direct impact on the result of the functional module clustering, the most suitable number of clusters is obtained by comparing the results of functional module clustering. The second phase of the proposed approach is cluster mapping. A suitable evaluation model for energy and delay optimization is presented in Subsection 3.2. A heuristic mapping algorithm called GMO is proposed to map the clusters onto the NoC for co-optimization of communication energy and delay. During iterative optimization, the mapping results are evaluated by the energy and delay model, and the best mapping scheme is selected as the final mapping result. The proposed approach can partition the SoC into small ICs of the same or near-same sizes, save power consumption, and reduce communication delays effectively.

3 Problem formulation and definitions

3.1 Definitions

The partitioning problem is how to partition the SoC into small ICs of equal areas and minimize communication energy and delays simultaneously. To formulate the problem, we refer to [10–12] and define the following terminologies.

Definition 1 (The functional module characteristic graph (FMCG)). The functional modules of SoC are modeled as a directed graph $G(\mathrm{FM}, A)$, where each vertex $\mathrm{fm}_i \in \mathrm{FM}$ represents a functional module, each edge $a_{ij} \in A$ represents the communication between fm_i and fm_j, and the weight of each edge v_{ij} indicates the communication volume on edge a_{ij}.

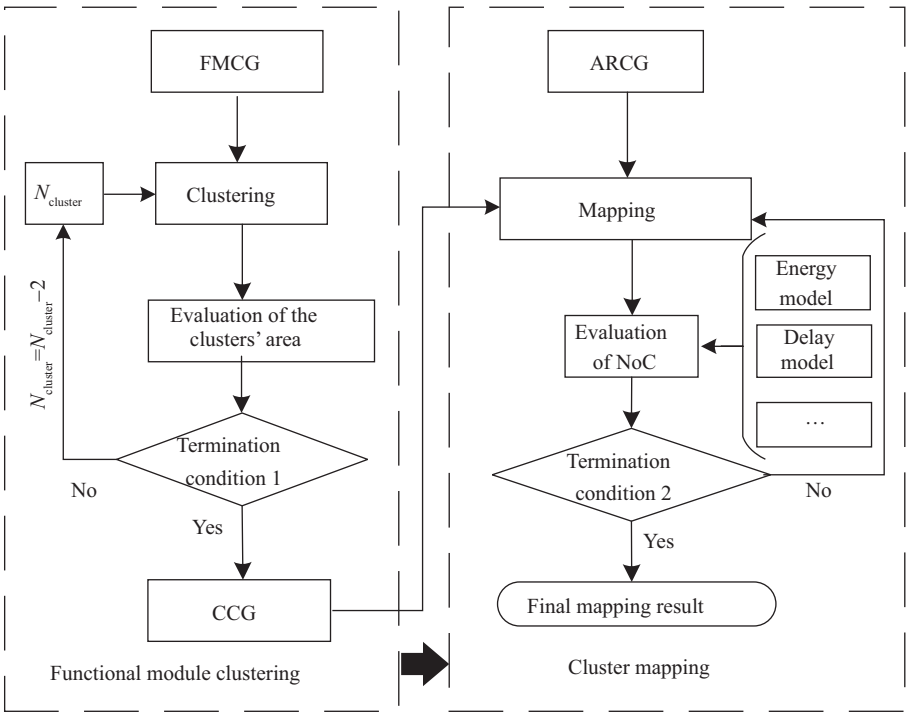

Figure 5 Overview of the proposed partitioning approach of the SoC.

Definition 2 (The cluster communication graph (CCG)). The cluster is modeled as a directed graph $G(C, F)$, where each vertex $c_i \in C$ represents a cluster which consists of a set of functional modules, each edge $f_{ij} \in F$ represents the traffic flow from c_i to c_j, and the weight of each edge f_{ij} indicates the communication volume on edge f_{ij}.

Definition 3 (The architecture characterization graph (ARCG)). The NoC architecture is modelled as a directed graph $G(R, L)$, in which nodes of the graph represent routers $r_i \in R$ and the edges between nodes $l_{ij} \in L$ represent physical links between routers. The bandwidth of the link between router r_i and r_j which is the weight of the edge l_{ij}, is denoted by $B(l_{ij})$.

Figure 6 shows examples of the FMCG, CCG, and ARCG. A simple FMCG is shown in Figure 6(a). The red dotted circles represent clusters of functional modules. Figure 6(b) shows a simple CCG with nine clusters. The traffic flows between clusters are calculated based on the communication volume between functional modules. Figure 6(c) shows the ARCG of a 2D mesh topology. NI represents the network interface which is used to convert data according to the transport protocol. c_i refers to a cluster of functional modules.

3.2 Objective function

The objective of partitioning the SoC is to balance the areas of partitioned ICs and reduce communication energy and delay simultaneously. The proposed approach has two phases: functional module clustering and cluster mapping. The objective function of functional module clustering is

$$f_1 = \left(\sum_{i=1}^{p} S(c_i) - \frac{\sum_{j=1}^{q} S(\mathrm{fm}_j)}{p}\right)^2 \Big/ p, \tag{1}$$

where $S(c_i)$ and $S(\mathrm{fm}_j)$ represent the areas of cluster c_i and functional module fm_j, respectively. p and q are the numbers of clusters and functional modules, respectively. During cluster mapping, the objective function becomes

$$f_2 = 1/(\alpha \mathrm{NE}_c + (1-\alpha)\mathrm{NT}_l), \tag{2}$$

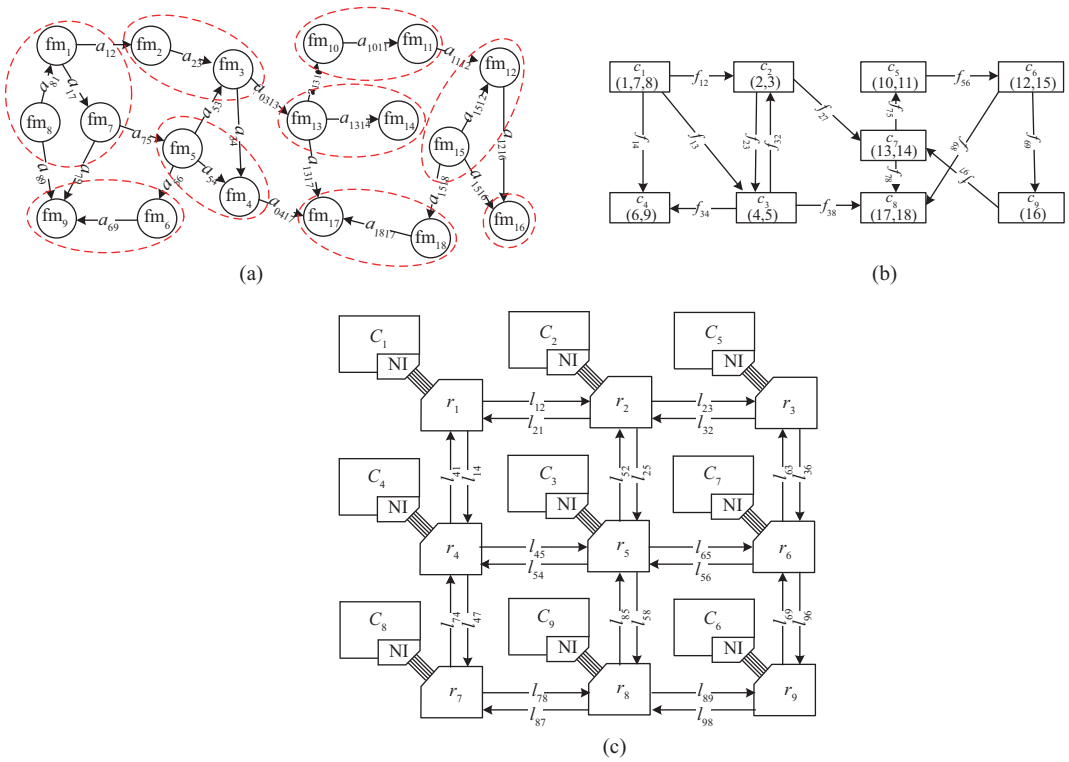

Figure 6 (Color online) Examples of (a) functional module characteristic graph, (b) cluster communication graph, and (c) architecture characterization graph.

where NE_c and NT_l are the normalized communication energy and delay, respectively. $\alpha \in [0,1]$ is a weight parameter that can adjust the weights of energy and delay.

To estimate the communication energy, the concept of bit energy [13] is introduced. E_{ij} represents the energy consumed by transporting one bit data from router r_i to r_j and is given by

$$E_{ij} = (\mathrm{hops}_{i,j} - 1) \times E_{\mathrm{lbit}} + \mathrm{hops}_{i,j} \times E_{\mathrm{Rbit}}, \tag{3}$$

where E_{lbit} and E_{Rbit} represent the energy consumed by transmitting one bit of data through a link and a router, respectively. $\mathrm{hops}_{i,j}$ is the number of hops that one bit of data takes from router r_i and r_j. Based on (3), the communication energy E_c is given by

$$E_c = \sum b_{ij} \times E_{ij}. \tag{4}$$

Here, b_{ij} is the communication volume between router r_i and r_j. To calculate the communication delay, the average network delay T_{av} [14] is introduced,

$$T_{\mathrm{av}} = \frac{T_l + T_r}{\sum_{i=1}^{n_c} \sum_{j=1}^{n_c} \lambda_{i,j}}, \tag{5}$$

where T_l is the delay from the links and T_r is the delay from the routers, $\lambda_{i,j}$ represents the number of flits transmitted from cluster c_i to c_j, and n_c refers to the number of clusters.

4 Partitioning algorithm

The main idea of our proposed approach is to partition the SoC reasonably to optimize the communication energy and delay. Our approach consists of two phases: (1) functional module clustering, and (2) cluster mapping.

4.1 Functional module clustering

In the first phase, the functional modules for a given SoC are divided into several clusters under the design constraint based on their communication requirements and their areas. Our algorithm is based on the greedy algorithm. The pseudo code of the clustering algorithm is shown below (Algorithm 1).

We define a variable N_{cluster}, which denotes the number of clusters and a variable $S(\text{fm}_i)$ which denotes the area of the functional module fm_i. Based on $S(\text{fm}_i)$ and N_{cluster}, the average area of clusters S_{avg} is calculated and S_{avg} is set as the criterion determining for whether the cluster is full. According to the weight of each edge v_{ij} in the $G(\text{FM}, A)$, the functional modules are sorted in descending order, and the first one is set as the cluster head. If the area of the cluster $S(c_i)$ is less than that of S_{avg}, the cluster is considered not full. A functional module with has the largest communication volume with the cluster head is placed in the cluster. When no functional module is adjacent to the cluster head, a new cluster head is selected from those functional modules in the cluster. If $S(c_i)$ is equal to or larger than S_{avg}, the cluster is considered full and a new cluster is generated in the same way. After the clusters are generated, the communication volume between clusters is calculated and the cluster communication graph $G(C, F)$ is obtained.

Algorithm 1 Function module clustering

Input: The application characteristics graph (FMCG) of the given SoC $G(\text{FM}, A)$, the number of clusters N_{cluster}, and the area of the functional module $S(\text{fm}_i)$;
Output: A cluster communication graph (CCG) $G(C, F)$.
1: Calculate the average area of clusters S_{avg} based on N_{cluste};
2: Based on the communication volume, sort the functional modules in descending order;
3: Set the first functional module in the sort as cluster head;
4: **for** $i = 1$; $i < N_{\text{cluster}}$; $i++$ **do**
5: **while** $(S(c_i) < S_{\text{avg}})$ **do**
6: **if** ($\exists$ adjacent functional module $\notin c_i$) **then**
7: Put an adjacent functional module in c_i;
8: **else**
9: Select another functional module from c_i as the new cluster head;
10: **end if**
11: **end while**
12: Select a new cluster head from the rest of the functional modules;
13: **end for**
14: Calculate the communication volume between clusters F;
15: Obtain the CCG $G(C, F)$.

4.2 Cluster mapping

In the second phase, the clusters for the given SoC are mapped onto the NoC architecture for co-optimization of communication energy and delay. Because the mapping problem is an NP-hard problem [9], a heuristic algorithm called GMO is proposed to obtain a near-to-optimal mapping result. In Algorithm 2, an individual with a chromosome represents a single solution. To encode the solutions, permutation encoding is applied. Every chromosome is represented by a string of numbers called genes. The position of a gene represents the identifier of clusters, and the content of a gene represents the router to which the cluster is connected. The population represents a collection of individuals. For example, for a 3×3 mapping problem in which nine clusters are to be mapped onto a 3×3 mesh NoC, a feasible chromosome can be encoded as [5 3 1 6 9 2 7 8 4]. In the chromosome, each gene represents a cluster that is connected to a router. For instance, the first gene "5" means the cluster c_1 is connected to router r_5, and the second gene "3" means the cluster c_2 is connected to router r_3.

In the GMO algorithm, a parent population is generated at random and the fitness of an individual in the parent population is calculated. The fitness F is calculated based on (2), which is described in Subsection 3.2. Then, a set of genetic operators, which includes a selection operator, a crossover operator, and a mutation operator, is selected by roulette wheel selection. The operator pool, which includes three

Algorithm 2 Cluster mapping

Input: Cluster communication graph CCG, architecture characterization graph ARCG and routing algorithm;
Output: Position of mapped clusters.
1: Initialize the parent population;
2: Calculate the fitness of the parent population;
3: **while** (not termination condition) **do**
4: Select operators from the operator pool by roulette wheel selection;
5: Obtain the offspring population from the selected genetic operators;
6: Calculate the fitness of the offspring population;
7: Update the weights of operators based on the "reward" mechanism;
8: Compare the parent and offspring populations based on fitness;
9: Generate a new parent population;
10: **end while**
11: Obtain the best individual as the mapping result.

Table 1 Genetic operators in the operator pool

Selection	Crossover	Mutation
S1: Truncation selection	C1: Discrete recombination	M1: Random mutation
S2: Tournament selection	C2: Single point recombination	M2: Swap mutation
S3: Roulette wheel selection	C3: Multi-point recombination	M3: Discrete breeder mutation

selection operators, three crossover operators and three mutation operators, is shown in Table 1. The principle of these operators is described in [15] in detail. After selection, crossover, and mutation, the offspring population is obtained. If some individuals in the offspring population improve the fitness obtained, these individuals will replace those individuals with low fitness in the parent population, and a new parent population is generated. Based on the improvement in the offspring population's fitness, the weight of the operators is updated. The weight function f_{op} is shown as follows:

$$f_{\text{op}} = f_{\text{op}} + \beta \mathrm{e}^{1/(N_{\text{iter}}+1-n_{\text{iter}})} \times \frac{\max(F_i(n)) - \max(F_i(n-1))}{\max(F_i(n))}, \tag{6}$$

where $F_i(n)$ is the fitness of individual i at the n-th iteration, $F_i(n-1)$ is the fitness of individual i at the $(n-1)$-th iteration, β is a coefficient that can adjust the increase rate of f_{op}, N_{iter} is the total number of iterations, and n_{iter} means the number of iterations already completed. Because the difficulty in obtaining a better result increases with the number of iterations, we use a nonlinear increasing function $\mathrm{e}^{1/(N_{\text{iter}}+1-n_{\text{iter}})}$ to increase the weight of "reward" with the number of iterations. The "reward" mechanism helps suitable operators obtain larger opportunities for application in the iteration. After N_{iter} iterations, an individual with the largest fitness is obtained, and its chromosome is the final mapping result.

5 Experimental results

In this section, we implement our algorithm in MATLAB R2008a. The topology of the ARCG is 2D mesh, and the routing algorithm is the west-first X-Y dimension-ordered routing function, which sends packets along the X-dimension first, and then by the Y-dimension next. To avoid a deadlock, the routing algorithm forbids southwest and northwest turns in the clockwise and counter-clockwise cycles, respectively. Eq. (2), which is described in Subsection 3.2 is used to calculate fitness F in the cluster mapping algorithm. In the experiment, the weights of energy and delay are both set to 0.5. To calculate the energy consumption, the bit energy values for the link and router are set to 0.449 pJ, 4.171 pJ, respectively, according to [11]. Because the two phases of the partitioning approach are relatively independent, we test the proposed clustering algorithm first and then the cluster mapping algorithm.

To verify the efficiency of the proposed partitioning algorithm, several benchmarks are applied: SoC25 from [16], SoC26 from [17], and SoC38 from [18]. For comparison, two other larger benchmarks, SoC50 and SoC80, are generated by task graph for free [19]. To show the effectiveness of the proposed clustering

Table 2 Effectiveness of the proposed functional module clustering algorithm

Benchmark	V	E	No. of clusters	No clusters [3]			[20]				Proposed			
				VA	CV	Area	VA	CV	Area	t (s)	VA	CV	Area	t (s)
SoC25	25	35	8	7.05	822	225	5.91	535	138	0.05	2.94	309	130	0.01
SoC26	26	62	6	55.09	5480	624	55.47	2160	437	0.10	7.47	2320	396	0.04
SoC38	38	47	12	10.14	11892	608	18.70	4101	342	0.12	9.13	5053	336	0.04
SoC50	50	147	9	4.39	37457	425	6.44	27209	387	0.18	0.86	23099	360	0.07
SoC80	80	292	9	20.78	78992	1280	12.54	59325	1161	1.20	7.43	51332	1071	0.40

Table 3 Average runtime of the mapping algorithms

Benchmark	No. of clusters	SA [21]	PSA [22]	GA [23]	GMO
C_SoC25	8	11.26	14.06	6.19	1.13
C_SoC26	6	0.07	0.05	0.04	0.04
C_SoC38	12	67.81	84.66	108.12	27.38
C_SoC50	9	17.70	24.05	24.04	8.98
C_SoC80	9	23.18	33.35	23.45	8.93

algorithm, we compare our functional module clustering algorithm with the algorithm in [20] first, as shown in Table 2. The approach in [3] is that the SoC is partitioned into small ICs based on the functional module. The columns V and E refer to the numbers of vertices and edges in the FMCG, respectively. The column VA indicates the variance of the cluster area, the column Area indicates the total area of all clusters, and the column CV indicates the sum of the communication volume between clusters. Column t indicates the runtime that the algorithm takes to get the clusters of functional modules. Compared to [3], both clustering algorithms can reduce the communication volume between clusters. Because the VA is more important than CV during the clustering process, the priority of VA is higher than CV in our proposed clustering algorithm. The functional module with the larger communication volume with the cluster head may be placed in the adjacent cluster to reduce VA, which explains why the CV obtained by the proposed clustering algorithm is larger than that obtained by the algorithm in [20] when the benchmarks are SoC26 and SoC38. Although the algorithm in [20] can sometimes yield lower CV values, it fails to reduce the difference between cluster areas. Compared with the algorithm in [20], the proposed clustering algorithm can reduce differences between cluster areas effectively. Compared with [20], the variance of the cluster area is reduced by 40.75%–86.65% and the runtime is reduced by 60.00%–80.00%. As shown in Table 2, both clustering algorithms can reduce the total area of all clusters, but the proposed algorithm reveals a better performance. Compared with [3], the proposed algorithm can reduce the total area of all clusters by 15.29%–42.22%.

To verify the validity of the mapping algorithm we proposed, the simulated annealing algorithm [21], particle swarm algorithm [22], and genetic algorithm [23] are implemented, respectively. For fair comparison, the same power and delay models described in Subsection 3.2 are used. Because the mapping algorithm is used to map the partitioned SoC on the NoC architecture, the clustering results of SoC25, SoC26, SoC38, SoC50 and SoC80 are set as benchmarks. Table 3 shows the average runtime of different algorithms; here, the termination condition is set as the best mapping result which obtained after 10000 random mappings. Table 3 clearly shows that GMO takes less time to achieve the mapping results than that required by the three other algorithms, especially when the number of clusters is large. Table 4 illustrates the mapping results of the different algorithm after 1000 iterations. The columns E and D present improvements in communication energy and delay, respectively. Compared with random mapping, GMO can achieve 10.64%–56.63% communication energy savings and 15.06%–50.30% communication delay savings. When the number of clusters is less than nine, all four algorithms can achieve the best mapping results. When the number of clusters exceeds nine, however, GMO yields the best mapping result.

Table 4 Improvements in communication energy and delay achieved by the mapping algorithms

Benchmark	SA [21]		PSA [22]		GA [23]		GMO	
	E (%)	D (%)	E (%)	D (%)	E (%)	D (%)	E (%)	D (%)
C_SoC25	56.63	41.08	56.63	41.08	56.63	41.08	56.63	41.08
C_SoC26	24.28	36.62	24.28	36.62	24.28	36.62	24.28	36.62
C_SoC38	35.85	48.76	35.65	49.78	35.28	49.91	36.45	50.30
C_SoC50	12.91	16.35	13.34	16.60	12.78	15.60	13.91	18.35
C_SoC80	10.10	14.57	10.34	14.06	9.21	14.43	10.64	15.06

6 Conclusion

In this paper, we proposed a partitioning approach of the SoC to improve the bendability and stretchability of flexible and stretchable systems. The approach features two phases. In the first phase, the SoC is partitioned into clusters of functional modules according to the communication volume and area constraint. In the second phase, the clusters are mapped onto the 2D mesh NoC for co-optimization of communication energy and delay. Experimental results showed that our approach can effectively partition the SoC into small ICs of the same, or nearly the same, size, save power consumption by about 10.64%–56.63% and reduce communication delays by 15.06%–50.30% simultaneously.

Acknowledgements This work was supported by National Basic Research Program of China (Grant No. 2015CB351906), National Natural Science Foundation of China (Grant No. 61172030), and Programme of Introducing Talents of Discipline to Universities (111 Project) (Grant No. B12026).

References

1 Wong W S, Salleo A. Flexible Electronics: Materials and Applications. Berlin: Springer, 2009
2 Sekitani T, Zschieschang U, Klauk H, et al. Flexible organic transistors and circuits with extreme bending stability. Nat Mater, 2010, 9: 1015–1022
3 Gupta U, Ogras U Y. Extending networks from chips to flexible and stretchable electronics. In: Proceedings of the 10th IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Nara, 2016. 1–6
4 Council N R. Flexible Electronics for Security, Manufacturing, and Growth in the United States: Summary of a Symposium. Washington: The National Academies Press, 2013
5 Kim D H, Ahn J H, Choi W M, et al. Stretchable and foldable silicon integrated circuits. Science, 2008, 320: 507–511
6 Kim D H, Song J, Choi W M, et al. From the cover: materials and noncoplanar mesh designs for integrated circuits with linear elastic responses to extreme mechanical deformations. Proc Natl Acad Sci USA, 2008, 105: 18675–18680
7 Yoon J, Baca A J, Park S I, et al. Ultrathin silicon solar microcells for semitransparent, mechanically flexible and microconcentrator module designs. Nat Mater, 2008, 7: 907–915
8 Kim D H, Kim Y S, Wu J, et al. Flexible electronics: ultrathin silicon circuits with strain-isolation layers and mesh layouts for high-performance electronics on fabric, vinyl, leather, and paper. Adv Mater, 2009, 21: 3703–3707
9 Fusella E, Cilardo A. PhoNoCMap: an application mapping tool for photonic networks-on-chip. In: Proceedings of 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016. 289–292
10 Chen W, Deng C C, Liu L B, et al. An efficient application mapping approach for the co-optimization of reliability, energy, and performance in reconfigurable NoC architectures. IEEE Trans Comput-Aided Des Integr Circ Syst, 2015, 34: 1264–1277
11 Li J S, Pan Y. A fast and energy efficient branch and bound algorithm for NoC task mapping. In: Proceedings of the 33rd IEEE International Conference on Computer Design (ICCD), New York, 2015. 9–16
12 Bo H, Song C, Wei Z, et al. Application-specific network-on-chip synthesis with topology-aware floorplanning. In: Proceedings of the 25th Symposium on Integrated Circuits and Systems Design (SBCCI), Brasilia, 2012. 1–6
13 Ye T T, Benini L, Micheli G D. Analysis of power consumption on switch fabrics in network routers. In: Proceedings of 2002 Design Automation Conference, New Orleans, 2002. 524–529
14 Elmiligi H, Morgan A A, El-Kharashi M W, et al. A delay-aware topology-based design for networks-on-chip applications. In: Proceedings of the 4th International Design and Test Workshop (IDT), Riyadh, 2009. 1–5

15 Sivanandam S N, Deepa S N. Introduction to Genetic Algorithms. Berlin: Springer, 2008
16 Chatha K S, Srinivasan K. Layout aware design of mesh based NoC architectures. In: Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS' 06), Seoul, 2006. 136–141
17 Seiculescu C, Murali S, Benini L, et al. SunFloor 3D: a tool for networks on chip topology synthesis for 3-D systems on chips. IEEE Trans Comput-Aided Des Integr Circ Syst, 2010, 29: 1987–2000
18 Yu B, Dong S Q, Chen S, et al. Floorplanning and topology generation for application-specific network-on-chip. In: Proceedings of the 15th Asia and South Pacific Design Automation Conference (ASP-DAC), Taipei, 2010. 535–540
19 Dick R P, Rhodes D L, Wolf W. TGFF: task graphs for free. In: Proceedings of Hardware/Software Codesign, Seattle, 1998. 97–101
20 Shuang Y, Fen G, Gui F, et al. A two-phase floorplanning approach for application-specific network-on-chip. In: Proceedings of the 10th International Conference on ASIC, Shenzhen, 2013. 1–4
21 Zhong L L, Sheng J Y, Jing M E, et al. An optimized mapping algorithm based on simulated annealing for regular NoC architecture. In: Proceedings of the 9th IEEE International Conference on ASIC, Xiamen, 2011. 389–392
22 Li Z X, Liu Y, Cheng M S. Solving NoC mapping problem with improved particle swarm algorithm. In: Proceedings of the 6th International Conference on Advanced Computational Intelligence (ICACI), Hangzhou, 2013. 12–16
23 Palaniveloo V A, Ambrose J A, Sowmya A. Improving GA-based NoC mapping algorithms using a formal model. In: Proceedings of 2014 IEEE Computer Society Annual Symposium on VLSI, Tampa, 2014. 344–349