Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7494829, 13 pages
https://doi.org/10.1155/2018/7494829

Research Article

Energy and Delay Optimization of Heterogeneous Multicore Wireless Multimedia Sensor Nodes by Adaptive Genetic-Simulated Annealing Algorithm

Xing Liu,[1,2] **Haiying Zhou**,[3] **Jianwen Xiang**,[1] **Shengwu Xiong**,[1] **Kun Mean Hou**,[2] **Christophe de Vaulx**,[2] **Huan Wang**,[1] **Tianhui Shen**,[1] **and Qing Wang**[1]

[1]*Hubei Key Laboratory of Transport Internet of Things, School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China*
[2]*LIMOS Laboratory, UMR 6158 CNRS (Centre National de la Recherche Scientifique), Clermont-Ferrand, France*
[3]*School of Electrical & Information, Hubei University of Automotive Technology, Shiyan, China*

Correspondence should be addressed to Shengwu Xiong; xiongsw@whut.edu.cn

Received 1 September 2017; Accepted 17 October 2017; Published 22 January 2018

Academic Editor: Jun Huang

Energy efficiency and delay optimization are significant for the proliferation of wireless multimedia sensor network (WMSN). In this article, an energy-efficient, delay-efficient, hardware and software cooptimization platform is researched to minimize the energy cost while guaranteeing the deadline of the real-time WMSN tasks. First, a multicore reconfigurable WMSN hardware platform is designed and implemented. This platform uses both the heterogeneous multicore architecture and the dynamic voltage and frequency scaling (DVFS) technique. By this means, the nodes can adjust the hardware characteristics dynamically in terms of the software run-time contexts. Consequently, the software can be executed more efficiently with less energy cost and shorter execution time. Then, based on this hardware platform, an energy and delay multiobjective optimization algorithm and a DVFS adaption algorithm are investigated. These algorithms aim to search out the global energy optimization solution within the acceptable calculation time and strip the time redundancy in the task executing process. Thus, the energy efficiency of the WMSN node can be improved significantly even under strict constraint of the execution time. Simulation and real-world experiments proved that the proposed approaches can decrease the energy cost by more than 29% compared to the traditional single-core WMSN node. Moreover, the node can react quickly to the time-sensitive events.

1. Introduction

Modern technological advances prompt the emergence of small-size, low-cost, and high-resolution visual information collection modules, which results in the rapid development of wireless multimedia sensor network (WMSN). WMSN is composed of a set of wirelessly interconnected tiny smart devices which retrieve the multimedia data pervasively from the surrounding environment and transmit them to the destinations wirelessly. Nowadays, WMSN technique is applied widely in diverse application domains and plays a significant role in the daily life [1].

Energy efficiency and delay optimization are critical and challenging issues for the proliferation of the WMSN. On the one hand, WMSN nodes are commonly powered by the energy-limit batteries and are deployed in remote and inaccessible areas where it is inconvenient or impossible to recharge the nodes. Thus, the nodes need to utilize the residual energy efficiently so as to keep a long lifetime. On the other hand, real-time responsiveness is required by many WMSN applications, for example, the industrial engine control. In these applications, the missing of the deadline can cause serious disasters. Therefore, the delay optimization is essential to ensure that the time-sensitive tasks can be completed within the deadline.

In the past, many research works were done to optimize the energy and delay of the WMSN. Data compression [2–4], data aggregation [5, 6], topology control [7, 8], energy-aware

protocol [9–15], data prediction [16], and sink mobility [17, 18] were commonly applied to conserve the energy, while delay-aware routing [10, 19] and delay-aware MAC [20, 21] were widely used to optimize the response delay. These past works improved the energy and delay performance of the WMSN significantly. However, some limitations still exist: (1) Most of the past research works focused on the software optimization works but ignored the hardware optimization strategies. However, without the hardware cooptimization, the energy and delay challenges cannot be addressed completely; for example, there are diverse WMSN applications, but most WMSN hardware platforms are not reconfigurable. If the hardware characteristic does not match the software features, the software cannot be executed efficiently even if the software optimization strategy is applied. Thus, it is essential to investigate an energy-efficient, delay-efficient, and reconfigurable hardware architecture in WMSN. (2) Most past works focused on reducing the energy cost of the wireless transmission but ignored the energy cost of the microcontrollers. This is because the wireless transmission is commonly considered to be high energy cost, yet there exist many cases in which the energy cost of the microcontrollers is also high and cannot be neglected; for example, in the plant disease monitoring application, the nodes capture the leaf image and perform signal processing to judge whether the plants have an infected disease or not. If yes, the image packet will be transmitted to the sink. Otherwise, no packet will be transmitted. In this kind of applications, the predominant energy cost derives from the data collection and signal processing on the WMSN microcontrollers, rather than the wireless transmission. In these cases, the energy optimization of the microcontrollers is critical. (3) Energy conservation and delay optimization are both significant to the WMSN. However, many past works did research on these two issues independently. As the development of WMSN, the research of an energy and delay multiobjective optimization mechanism becomes indispensable.

Regarding the research limitations above, the work in this article aims to investigate an energy-efficient, delay-efficient, hardware and software integrated optimization platform which can minimize the energy cost of the WMSN node while guaranteeing the deadline of the real-time WMSN tasks. Frist, a new energy-efficient, delay-efficient, reconfigurable WMSN hardware architecture is designed. This architecture uses both the heterogeneous multicore technique and the DVFS technique. By these techniques, the WMSN nodes can adapt the hardware characteristics dynamically in terms of the software contexts. Consequently, the software can be executed efficiently with less energy cost and shorter execution time. Second, an energy and delay multiobjective optimization algorithm is investigated. This algorithm incorporates the genetic algorithm with the simulated annealing algorithm so that the global optimization solution can be reached within acceptable calculation time. With this algorithm, the energy cost of the WMSN node can be decreased at the most while the deadline of the time-sensitive tasks can also be met. By combining the above hardware architecture and software optimization strategies, both the energy efficiency and the delay performance of the WMSN node can be promoted significantly.

The works in this article are based on two general assumptions. One is that the tasks can be duplicated to each microcontroller, which means each task can be executed on any microcontroller equipped on the WMSN node. The other is that the task model is deterministic; that is, the execution time and the relationship between the tasks are preknown.

The rest of this paper is organized as follows. Section 2 introduces the overview of the related works. Section 3 designs a new energy-efficient, delay-efficient, reconfigurable heterogeneous multicore WMSN hardware architecture. With this architecture, the software algorithms presented in Sections 4 and 5 can achieve better optimization performance. Section 4 builds the energy and delay models of heterogeneous multicore system and investigates an energy and delay multiobjective optimization algorithm GASA. Section 5 presents the DVFS adaption algorithm which can strengthen the performance of WMSN further on the basis of GASA optimization results. Section 6 evaluates the optimization performance by simulation. Section 7 implements a real-world multicore WMSN node in terms of the hardware architecture proposed in Section 4 and then evaluates its performance by real-world experiments. Section 8 concludes this article.

2. Related Works

Energy cost of WMSN can be optimized by many mechanisms such as the data compression, data aggregation, topology control, energy-aware protocol, data prediction, and sink mobility. Data compression is critical to conserve the energy as it can reduce the packet size and the wireless transmission time. Since the computational capability and communication bandwidth in the WMSN is constrained, specific audio and image compression techniques need to be developed for the WMSN devices [2–4]. Data aggregation technique enables the nodes located along a path towards the sink to perform the data fusion. It decreases the amount of data forwarded in the WMSN and results in lower energy cost [5, 6]. Topology control technique controls over some parameters of the network such as the transmission power of the nodes and the role of the nodes. By modifying these network parameters, least nodes can be selected to maintain the connectivity of topology, whereas the other nodes can fall asleep to conserve the energy [7, 8]. Energy-aware protocols are popularly researched in the WMSN, including the energy-aware routing, energy-efficient MAC, and cross-layer protocol. Energy-aware routing achieves the energy conservation by reducing the routing hops or decreasing the broadcast count [9, 10]. Energy-efficient MAC protocols lower the energy waste by decreasing the count of retransmissions, minimizing the interference, or maximizing the concurrency and reliability [11, 12]. Cross-layer protocols improve the communication performance by merging the information of different layers jointly; for example, the network layer can decide how to route by incorporating the congestion information from the transport layer and the link quality value from the MAC layer. This incorporation of different layers improves the

communication quality and maximizes the network lifetime [13–15]. Data prediction technique builds the prediction model which describes the data evolution of the sensed phenomenon within certain error bounds. It enables the user to acquire the information of the sensor nodes directly from the sink by prediction. It decreases the amount of data to be transmitted and optimizes the transmission energy cost [16]. Sink mobility is the technique which uses a mobile base station moving around the network to collect the node information. It balances the load of the WMSN nodes and prevents the nodes located closely to the sink from depleting the energy quickly. It can be used on the energy-constrained WMSN nodes to prolong the lifetime of the whole network [17, 18]. Delay optimization is commonly realized by the delay-aware routing and delay-aware MAC protocols in the WMSN. Delay-aware routing optimizes the delay by searching for the shortest path to the destination [10, 19], while delay-aware MAC protocol decreases the end-to-end delay by reducing the collision and packet retransmission [20, 21].

3. Energy-Efficient and Delay-Efficient Hardware Architecture Design

In this section, the design of an energy-efficient, delay-efficient, and reconfigurable WMSN hardware architecture is presented. Based on this hardware architecture, the WMSN hardware and software can match each other better. Consequently, the software optimization schemes can take effect more efficiently.

As the development of WMSN technique, the WMSN applications become more and more diverse. However, most WMSN hardware platforms are still inflexible that they cannot address the challenge of application diversity in WMSN. On the one hand, most WMSN hardware platforms are single-core architecture with only one type of microcontroller equipped. One type microcontroller can be efficient to run several kinds of applications but may not be efficient when running the other kinds. On the other hand, most WMSN hardware platforms are not reconfigurable. They cannot adjust their characteristics dynamically in terms of the software contexts. As a result, the software cannot be executed efficiently, and this degrades the energy efficiency and delays performance of the WMSN nodes.

One way to address the above challenge is to design a WMSN hardware platform which has the following features: (1) It equips more than one type of microcontrollers on the node. Each equipped microcontroller has differential specialties and is appropriate to run particular kinds of tasks. During the run-time, each software task can be assigned to the most appropriate microcontroller to be executed. By doing this, the software and hardware can match each other better. (2) It makes each microcontroller configurable, that is, enables the hardware to adjust its characteristic dynamically in terms of the software contexts. With the above mechanisms, the WMSN hardware can be highly context aware of the software running on it. Consequently, the energy and delay performance of the WMSN can be improved.

To realize the design concepts above, the heterogeneous multicore hardware architecture (Section 3.1) and the DVFS technique (Section 3.2) can be applied on the WMSN node.

3.1. Heterogeneous Multicore Design for Energy and Delay Optimization. Heterogeneous multicore hardware architecture can lower the energy cost and improve the delay performance if compared to the traditional single-core WMSN architecture.

(i) *Lower energy cost*: heterogeneous multicore platform is equipped with different types of microcontrollers. Since different microcontrollers have differential features and are good at executing a given kind of WMSN tasks [22–24], each WMSN task can be scheduled to the microcontroller which is the most energy efficient to run it. By doing this, the energy resources can be utilized more efficiently.

(ii) *Higher delay performance*: as several WMSN tasks can be executed concurrently by different microcontrollers in the multicore system, the completion time of the tasks can be shortened greatly if compared to the time cost of the traditional single-core system.

3.2. DVFS Technique for Context Awareness. DVFS is a power management technique with which the operating voltage and frequency of the hardware can be adjusted dynamically in terms of the performance requirement of the active application scenario [25, 26]. By decreasing the frequency, the executing time of the tasks will be prolonged linearly, whereas the energy cost can be reduced quadratically. Due to this functionality, the DVFS technique can be used in WMSN to conserve the energy under the delay constraint by balancing the tradeoff between the energy cost and the execution time: (1) If the real-time tasks cannot be completed before the deadline, the frequency of this microcontroller can be improved to shorten the execution time of these tasks. In this case, the delay performance is enhanced at the cost of lower energy efficiency. (2) If the real-time tasks are schedulable, the frequency of this microcontroller can be decreased gradually until the task completion time is near the deadline. In this case, the energy cost is reduced at the most under the constraint of the deadline. With the above adaption mechanism, the energy cost of the WMSN node can be optimized as much as possible. Moreover, the deadline of the time-sensitive tasks can also be guaranteed.

Figure 1 depicts the design diagram of the heterogeneous multicore WMSN platform. Several different types of microcontrollers are equipped. Tasks assigned to different microcontrollers can be executed concurrently, and the hardware characteristic of each microcontroller can be adjusted by the DVFS regulators during the run-time. The multiobjective optimization algorithm and the DVFS adaption algorithm can calculate each task's assignment and each microcontroller's working frequency. If no tasks are assigned to a microcontroller, this microcontroller can enter the ultralow-power idle status to save energy. If a microcontroller does not need to run any task for a long time, it can be powered off directly to avoid wasting the energy.

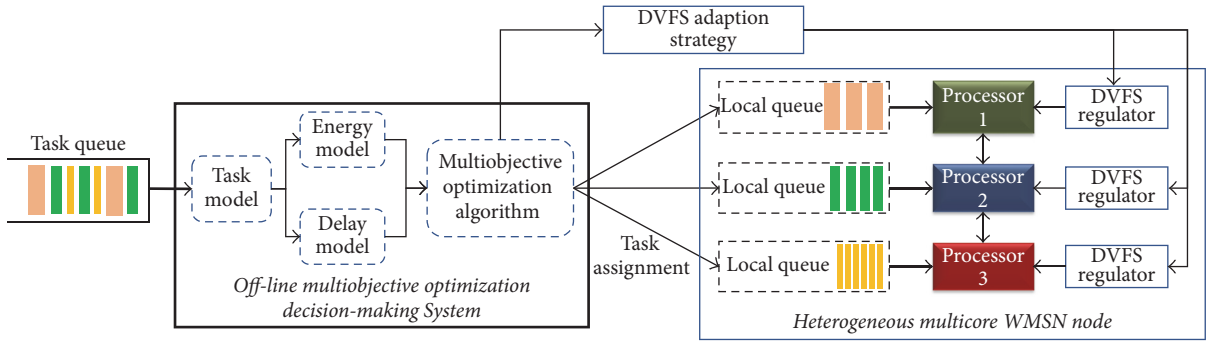

Figure 1: Elementary diagram of the energy-efficient, delay-efficient, context-aware heterogeneous multicore WMSN platform.

4. Energy and Delay Multiobjective Optimization Algorithm

In this section, we first build the energy and delay models of the heterogeneous multicore system in Section 4.1. Then, we state the energy and delay multiobjective optimization problem in Section 4.2. Finally, we investigate an energy and delay multiobjective optimization algorithm in Section 4.3.

4.1. Energy and Delay Models of Heterogeneous Multicore WMSN Node. The tasks on the heterogeneous multicore WMSN node can be modeled by the directed acyclic graph (DAG), where the vertexes represent the tasks and the edges represent the links between the tasks.

Define the parameters in DAG as follows:

(i) T_i $(i = 1, 2, \ldots, n)$ represents the WMSN tasks, where n is the task number.

(ii) P_j $(j = 1, 2, \ldots, m)$ represents the microcontrollers equipped on the multicore WMSN node, where m is the microcontroller number.

(iii) X_{ij} represents the multicore task scheduling results. If task T_i is assigned to the microcontroller P_j, X_{ij} will be 1. Otherwise, it will be zero.

(iv) L_{ij}^{tsk} represents the execution load of task T_i on microcontroller P_j. Its value depends on the software programming and the compiler.

(v) R_{ij}^{run} represents the energy consumption rate when executing task T_i on microcontroller P_j.

(vi) V_{ij}^{run} represents the execution speed when running task T_i on microcontroller P_j.

(vii) L_{ij}^{trans} represents the communication load between P_i and P_j. It will be zero if tasks assigned to P_i do not need to communicate with the tasks assigned to P_j.

(viii) R_{ij}^{trans} represents the energy consumption rate of the communication between P_i and P_j.

(ix) V_{ij}^{trans} represents the data transmission speed between P_i and P_j.

(x) R_j^{idle} represents the energy consumption rate of P_j when it is in idle status. Its value depends on the microcontroller hardware characteristic.

Based on the task model, the energy and delay models of the heterogeneous multicore WMSN system can be built.

4.1.1. Energy Model of Heterogeneous Multicore WMSN Node. Energy cost consists of three parts: one is the E^{run} which is consumed when the microcontroller is active, one is the E^{idle} which is consumed when the microcontroller is idle, and the other is the E^{trans} which is consumed by the data transmission among the microcontrollers.

E^{run} can be expressed as

$$E^{\text{run}}\left(X_{ij}\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij} R_{ij}^{\text{run}} t_{ij}^{\text{run}} = \sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij} R_{ij}^{\text{run}} \times \frac{L_{ij}^{\text{tsk}}}{V_{ij}^{\text{run}}}, \quad (1)$$

where t_{ij}^{run} represents the execution time of task T_i on microcontroller P_j.

E^{idle} can be expressed as

$$\begin{aligned} E_{\text{idle}}\left(X_{ij}\right) &= \sum_{j=1}^{m} R_j^{\text{idle}} t_j^{\text{idle}} \\ &= \sum_{j=1}^{m} R_j^{\text{idle}} \times \left(t^{\text{end}} - \sum_{i=1}^{n} X_{ij} t_{ij}^{\text{run}}\right) \\ &= \sum_{j=1}^{m} R_j^{\text{idle}} \times \left(t^{\text{end}} - \sum_{i=1}^{n} X_{ij} \times \frac{L_{ij}^{\text{tsk}}}{V_{ij}^{\text{run}}}\right), \end{aligned} \quad (2)$$

where t_j^{idle} represents the idle time of microcontroller P_j, and t^{end} represents the task completion time. Task completion time can be calculated by the time model in (7).

E^{trans} can be expressed as

$$E^{\text{trans}}\left(X_{ij}\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{ij}^{\text{trans}} t_{ij}^{\text{trans}} = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{ij}^{\text{trans}} \times \frac{L_{ij}^{\text{trans}}}{V_{ij}^{\text{trans}}}. \quad (3)$$

In terms of (1) to (3), the total energy cost of the heterogeneous multicore WMSN node can be expressed as

$$E\left(X_{ij}\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}R_{ij}^{\text{run}} \times \frac{L_{ij}^{\text{tsk}}}{V_{ij}^{\text{run}}} + \sum_{j=1}^{m} R_{j}^{\text{idle}} \times \left(t^{\text{end}} - \sum_{i=1}^{n} X_{ij}\frac{L_{ij}^{\text{tsk}}}{V_{ij}^{\text{run}}}\right) + \sum_{i=1}^{n}\sum_{j=1}^{m} R_{ij}^{\text{trans}} \times \frac{L_{ij}^{\text{trans}}}{V_{ij}^{\text{trans}}}. \tag{4}$$

4.1.2. Delay Model of Heterogeneous Multicore WMSN Node. Suppose the task set of the multicore WMSN node Q is expressed as follows:

$$Q = Q_{\text{seq}} \cup Q_{\text{con}}, \tag{5}$$

where Q_{seq} represents the set of the tasks which should be executed in sequence and Q_{con} represents the set of the tasks which can be executed concurrently on different microcontrollers.

Based on the task scheduling result X_{ij}, the task set Q_{con} can be divided into a serial of subset as follows:

$$Q_{\text{con}} = Q_{\text{con}}^{P_1} \cup Q_{\text{con}}^{P_2} \cup \cdots \cup Q_{\text{con}}^{P_m}, \tag{6}$$

where $Q_{\text{con}}^{P_i}$ represents the tasks assigned to the microcontroller P_i.

In terms of (5) and (6), the time model of running task on the heterogeneous multicore WMSN node can be expressed as follows:

$$t = t\left(Q_{\text{seq}}\right) + t\left(Q_{\text{con}}\right) = t\left(Q_{\text{seq}}\right) + \max\left(t\left(Q_{\text{con}}^{P_1}\right), t\left(Q_{\text{con}}^{P_2}\right), \ldots, t\left(Q_{\text{con}}^{P_m}\right)\right), \tag{7}$$

where $t(Q_{\text{seq}})$ and $t(Q_{\text{con}}^{P_j})$ represent the time of running the task set Q_{seq} and task set $Q_{\text{con}}^{P_j}$, respectively; for example, if $Q_{\text{con}}^{P_j}$ is equal to $\{T_1, T_2, \ldots, T_k\}$, then $t(Q_{\text{con}}^{P_j})$ will be as follows:

$$t\left(Q_{\text{con}}^{P_j}\right) = \sum_{i=1}^{k}\left(\frac{L_{ij}^{\text{run}}}{V_{ij}^{\text{run}}} + \frac{L_{ij}^{\text{trans}}}{V_{ij}^{\text{trans}}}\right). \tag{8}$$

4.2. Problem Statement. The research objective is to minimize both the energy cost (i.e., (4)) and the execution time (i.e., (7)) of the WMSN node, and this is a multiobjective optimization problem. To simplify the calculation complexity of the multiobjective optimization problem, we convert it to the single objective optimization problem by making the energy the predominant optimization objective whereas the time is the constraint, depicted as follows:

$$\begin{aligned} &\text{Minimize} \quad E\left(X_{ij}\right) = \sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}R_{ij}^{\text{run}} \times \frac{L_{ij}}{S_j} + \sum_{j=1}^{m} R_{j}^{\text{idle}} \times \left(t^{\text{end}} - \sum_{i=1}^{n} X_{ij}t_{ij}^{\text{run}}\right) + \sum_{i=1}^{n}\sum_{j=1}^{m} R_{ij}^{\text{trans}} \times \frac{L_{ij}^{\text{trans}}}{V_{ij}} \\ &\text{subject to} \quad t = t\left(Q_{\text{seq}}\right) + \max\left(t\left(Q_{\text{con}}^{P_1}\right), t\left(Q_{\text{con}}^{P_2}\right), \ldots, t\left(Q_{\text{con}}^{P_m}\right)\right) < t^{\text{dln}}, \end{aligned} \tag{9}$$

where t represents the execution time of the tasks and t^{dln} represents the deadlines of the tasks.

4.3. Genetic-Simulated Annealing Algorithm for Energy and Delay Multiobjective Optimization. The multicore task scheduling problem is known as the NP-complete problem [27]. Since this problem is computationally intractable even if the assumptions are simplified [28], the heuristic algorithms such as the genetic algorithm (GA) and the simulated annealing (SA) algorithm are commonly applied to obtain the optimal and suboptimal solutions to this kind of problem.

GA is a metaheuristic inspired by the process of natural selection. It is commonly used to generate high-quality solutions to optimization and search problems by bioinspired operators such as mutation, crossover, and selection [29]. GA can obtain the optimization solution quickly, but it is prone to trap into the local optima. Fortunately, the SA algorithm, another metaheuristic to approximate global optimization of a given function in a large search space, has the ability to jump out of the local optimization and reaches the best optimization [30]. Thus, the GA is incorporated with SA in this article, named as GASA. By means of GASA, the global optima can be searched out within acceptable time.

The flow chart of the GASA algorithm is illustrated in Figure 2. First, the GA is applied to rapidly search for an optimal solution within the solution space. Then, the SA is used to further search for a better solution based on the GA result.

4.3.1. GA Encoding and Decoding. Each GA chromosome C_k ($k = 1, 2, \ldots, popSize$) represents one feasible multicore scheduling scheme, where $popSize$ is the number of chromosomes in the generation. Suppose n tasks $(T_1, T_2, \ldots, T_n)$ are to be executed on m microcontrollers $(P_0, P_1, \ldots, P_{m-1})$. Then, each chromosome can be encoded as $C_k = [S_1, S_2, \ldots, S_n]$, where S_i represents the microcontroller to which the task T_i is allocated. Since there are m microcontrollers, each S_i can be encoded to a $\log_2^m$ bit binary. Thus, the length of each chromosome will be $n \times \log_2^m$; for example, if n is five, m is four, and a chromosome is encoded as 00 10 01 10 11, then this chromosome represents tasks T_1 to T_5 assigned to microcontrollers P_0, P_2, P_1, P_2, P_3, respectively.

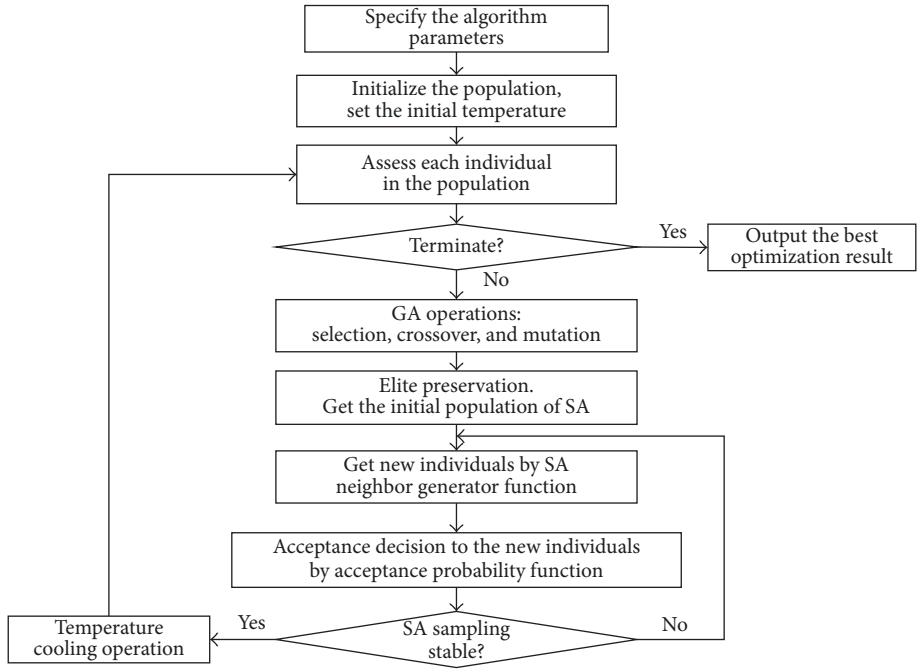

Figure 2: Flow chart of the GASA algorithm.

Table 1: (a) Original chromosome. (b) Crossover by exchanging the bottom halves. (c) Crossover by exchanging the top halves. (d) Mutation operation.

		T_1	T_2	T_3	T_4	T_5
(a)	Chromosome A	00	10	01	10	11
	Chromosome B	10	01	00	11	01
(b)	Chromosome A	00	10	00	11	01
	Chromosome B	10	01	01	10	11
(c)	Chromosome A	10	01	01	10	11
	Chromosome B	00	10	00	11	01
(d)	Chromosome A	10	**11**	01	10	11
	Chromosome B	00	10	00	11	01

4.3.2. GA Operations. To generate a new generation of the population, the GA crossover and mutation need to be operated. Crossover can be realized by exchanging portions of two chromosomes. First, select a crossover site which divides the chromosome into two halves. Then, exchange the bottom halves of the two selected chromosomes; for example, in Table 1(a), if the crossover site is task T_3, the new chromosomes after the crossover will be the ones shown in Table 1(b). However, if the exchange is always performed from the bottom halves, the top halves of the chromosomes will keep unchanged; for example, the genes related to tasks T_1 and T_2 in Table 1 ((a) and (b)) keep identical even if the crossover is performed. This indicates that some tasks are always assigned to the fixed microcontrollers, and this is not helpful to search for the optimal solution. To solve this problem, we change the crossover operation by operating the exchange either from the bottom halves Table 1(b) or from the top halves Table 1(c) of the chromosomes, and which way to be selected is random. Mutation can be performed by mutating one gene in the chromosome; for example, in Table 1(d), the gene of task T_2 in the chromosome A is mutated from "01" (representing P_1) to "11" (representing P_3).

4.3.3. GA Fitness Function and Constraint. Fitness function can measure the quality of a represented solution and determine the selection probability of this solution. As the predominant optimization objective of this article is the energy cost, the energy equation expressed in (9) is used as the fitness function. The smaller the energy value $E(X_{ij})$ is, the greater fitness the scheduling solution X_{ij} (represented by chromosome) will have.

However, the energy cost needs to be optimized under the time constraint, shown in (9). Thus, each solution should be checked to determine whether the constraint is met before assessing its fitness. If the constraint is not satisfied, two measures can be taken. One is to continue making crossover or mutation until the solution meets the constraint. The other is to use the penalty mechanism and lower the fitness value of this solution. Once the fitness value is lowered, the probability of passing this solution to the next generation will become smaller.

4.3.4. SA Neighbour Generator Function. SA uses the neighbour of a solution as a way to explore the solution space. Although it prefers better neighbour, it also accepts worse neighbours to avoid getting stuck in local optima. New neighbours in SA are generated by the neighbour generator function. In this article, the neighbour of an SA solution is

generated by exchanging the position of two genes in the chromosome; for example, one neighbour of the chromosome "10 11 01 10 11" can be "10 01 11 10 11."

5. DVFS Adaption to Further Strengthen the Optimization Performance

In Section 4, we have investigated a multiobjective optimization algorithm, GASA. In this section, the DVFS adaption algorithm is applied further on the basis of the GASA optimization result so as to strengthen the energy and delay optimization performance of the GASA.

The principle of the DVFS energy conservation mechanism can be depicted briefly as follows: $E = P \times T$, where E represents the energy cost, P represents the power, and T represents the time. When the frequency of the microcontroller is decreased by the DVFS regulator, the time T will be prolonged linearly whereas the power P can be declined cubically. Consequently, the energy cost E can be reduced quadratically [25]. Thus, to conserve the energy at the most, the frequency of the microcontroller can be turned down gradually as long as the execution time does not exceed the deadline. By this means, the energy cost becomes minimum, yet the deadline of the real-time tasks can also be guaranteed.

Based on the principle above, the DVFS adaption mechanism is investigated. Suppose the multicore task scheduling status calculated by the GASA algorithm is as in Figure 3:

(i) There are two microcontrollers P_1, P_2 and six WMSN tasks $T_1, T_2, \ldots, T_6$. E_i represents the energy cost of task T_i, and t_i represents the execution time of task T_i.

(ii) Tasks T_2 and T_3 can run concurrently. Tasks T_5 and T_6 can run concurrently.

(iii) In terms of the scheduling result calculated by GASA, tasks T_1, T_2, and T_5 are assigned to microcontroller P_1 while the others are assigned to microcontroller P_2.

(iv) The completion time of all tasks is T_{end}. Deadline for all tasks is T_{dedl}.

From Figure 3, it can be seen that some slack periods exist during the task execution process. One is the slack time between the completion of task T_3 and the completion of task T_2, one is the slack time between the completion of task T_5 and the completion of task T_6, and the other is the slack time between the ending time T_{end} and the deadline time T_{dedl}.

The slack time indicates the time redundancy, and this redundancy can be stripped by the DVFS technique through adjusting the working frequency of the microcontrollers. With the adjustment, the energy cost of the WMSN can be optimized further. After the adjustment, the task execution process illustrated in Figure 3 will change to the process shown in Figure 4. On the one hand, the average working frequency of the tasks is lowered until the task completion time T_{end} equals the deadline T_{dedl}. On the other hand, the frequencies of tasks T_3 and T_5 are decreased until their completion time is near to that of tasks T_2 and T_6, respectively.

Mathematically, the DVFS adaption problem can be stated as follows:

$$\begin{aligned} \text{minimize} \quad & E(k_i) = \frac{1}{k_1^2} \times E_1 + \frac{1}{k_2^2} \times E_2 + \frac{1}{k_4^2} \times E_4 + \frac{1}{k_6^2} \times E_6 + \left(\frac{t_3}{t_2}\right)^2 \times \frac{1}{k_2^2} \times E_3 + \left(\frac{t_5}{t_6}\right)^2 \times \frac{1}{k_6^2} \times E_5 \\ \text{subject to} \quad & T_{\text{end}}(k_i) = k_1 \times t_1 + k_2 \times t_2 + k_4 \times t_4 + k_6 \times t_6 \le T_{\text{dedl}}, \end{aligned} \tag{10}$$

where t_i ($i = 1, 2, \ldots, 6$) and E_i ($i = 1, 2, \ldots, 6$) are constants and k_i ($i = 1, 2, \ldots, 4$) is the adaption coefficient of the frequency. Since the frequency of the microcontrollers can be adjusted to only a set of scales by the DVFS regulators, the variable k_i is a set of discrete values rather than successive values. We also use the heuristic algorithm GASA presented in Section 4 to search for the optimal solution to this problem, yet a different GA encoding mechanism should be used. In this problem, the GA chromosome will no longer represent the multicore task scheduling strategy, but it represents the frequency scale of the microcontrollers. Suppose there are six tasks $(T_1, T_2, \ldots, T_6)$ and two microcontrollers (P_1, P_2), tasks T_3, T_4 and, T_6 are assigned to microcontroller P_1, tasks T_1, T_2, and T_5 are assigned to microcontroller P_2, microcontroller P_1 has four frequency scales $(F_{10}, F_{11}, \ldots, F_{13})$, and microcontroller P_2 has four frequency scales $(F_{20}, F_{21}, \ldots, F_{23})$. If the working frequencies of tasks $(T_1, T_2, \ldots, T_6)$ are, respectively, $(F_{20}, F_{22}, F_{11}, F_{13}, F_{22}, F_{13})$, then the chromosome will be encoded as "00 10 01 11 10 11."

6. Performance Evaluation by Simulation

In this section, we evaluate the optimization performance of GASA algorithm and the GASA plus DVFS algorithm (named as GASA-DVFS) by comparing them with the GA algorithm proposed by Monnier et al. [31] (named as M-GA) and the GA algorithm proposed by Theys et al. [32] (named as T-GA). The M-GA is designed for the real-time tasks on the homogeneous multicore system. The T-GA is designed for the general tasks on the heterogeneous multicore system.

A task graph example is needed to perform the evaluation, we use the method proposed in [33] to generate a random task graph. To perform this generation, several input parameters should be set, including the task number α, the shape parameter of the graph β, the average computation cost δ, the average communication cost ε, the communication to computation ratio γ, and the range percentage factor of the computation cost on the microcontrollers λ. These parameters are set as follows in our simulation works: $\alpha = 100$,

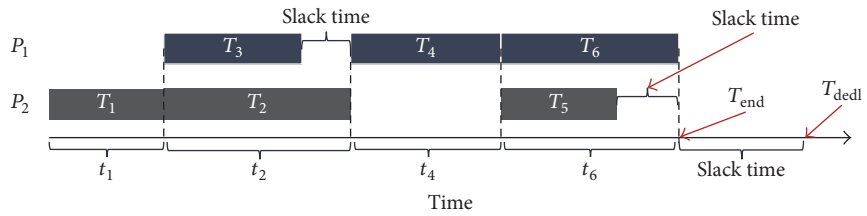

Figure 3: Multicore task scheduling process after the GASA algorithm is applied.

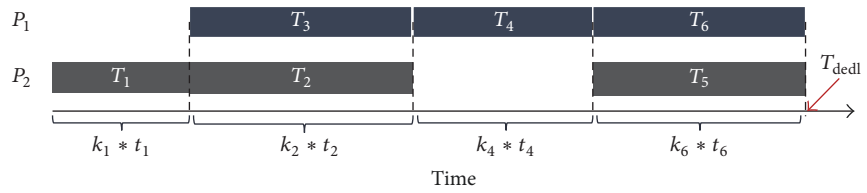

Figure 4: Task executing process after the DVFS adaption mechanism is performed.

$\beta = 0.4$, $\delta = 50$, $\varepsilon = 10$, $\gamma = 0.2$, $\lambda = 0.4$. In addition, the load of the task is set to a random value between 500 and 5000, and the execution speed of the microcontroller is set to a random value between 2000 and 16000.

The initial parameters should be set for the algorithms. We set 0.7 for the crossover, 0.1 for the mutation, 20 for the population size, 1500 for the maximum iteration count, 500 for the initial temperature, 0.98 for the cooling rate, and 4 for the microcontroller number. Suppose each microcontroller can work with 4 different frequency scales, and these scales are 0.9, 1, 1.1, and 1.2, respectively.

6.1. Performance Evaluation of Different Algorithms. We tested the energy cost and execution time of different algorithms and the result is shown in Figure 5. The result shows the following: (1) The DVFS technique can further strengthen the energy optimization performance of the GASA; for example, the energy cost by using the GASA in Figure 5(a) is 6900 mJ, yet it is 5660 mJ after the DVFS algorithm is applied. (2) GASA can achieve better energy efficiency than M-GA. This is because GASA incorporates the GA algorithm with the SA algorithm so that a global optimization solution is more likely to be searched out. (3) The GASA energy optimization performance decreases rapidly when the deadline constraint is strict. This is because some energy optimal solutions cannot be passed to the next generation in this case due to the missing of the deadline. (4) The DVFS algorithm has better optimization performance when the deadline constraint is not so strict. This is because more slack time exists in this case. The more the slack time is, the better the DVFS optimization performance will be. (5) T-GA is not a real-time scheduling algorithm. Thus, the deadline cannot be met by its calculation result.

6.2. Evolution Process of Energy Cost by GASA. To inspect the energy optimization evolution process of GASA, we calculate the average energy cost of each GASA iteration and illustrates them as shown in Figure 6. We can see that the energy cost is gradually reduced on the whole. There exist several periods when the GA algorithm is prone to trap into the local optima. However, the SA algorithm jumps out these traps successfully. Nearly after the 1200th iteration, the evolution process tends to be stable.

6.3. Performance Evaluation by Using Different Microcontrollers. To analyze the relationship between the microcontroller characteristics and the energy optimization efficiency, we compared the optimization results of the 4-core heterogeneous system, 8-core heterogeneous system, and 8-core homogeneous system, and the result is illustrated in Figure 7. From this result, we can see that the task execution time is generally decreased as the microcontroller number increases, and this is because the tasks can be executed concurrently on more microcontrollers. Compared to the 8-core homogeneous system, the 8-core heterogeneous system has better performance in this example. This is because more kinds of cores are equipped on the heterogeneous system. As a result, the hardware platform can be more context aware of the software, and the software can be executed more efficiently with less energy and time cost.

7. Real-World Implementation and Evaluation

In this section, the real-world implementation and evaluation works of a heterogeneous multicore WMSN node designed for the fire detection application are presented.

7.1. Implementation of the Heterogeneous Multicore WMSN Node. The application scene of the fire detection is as follows: the WMSN node detects the environment temperature and gas concentration continuously by the temperature and gas sensors. In case the temperature is increased abnormally and the gas concentration is improved rapidly, the WMSN node opens the camera to capture the fire scene, processes the

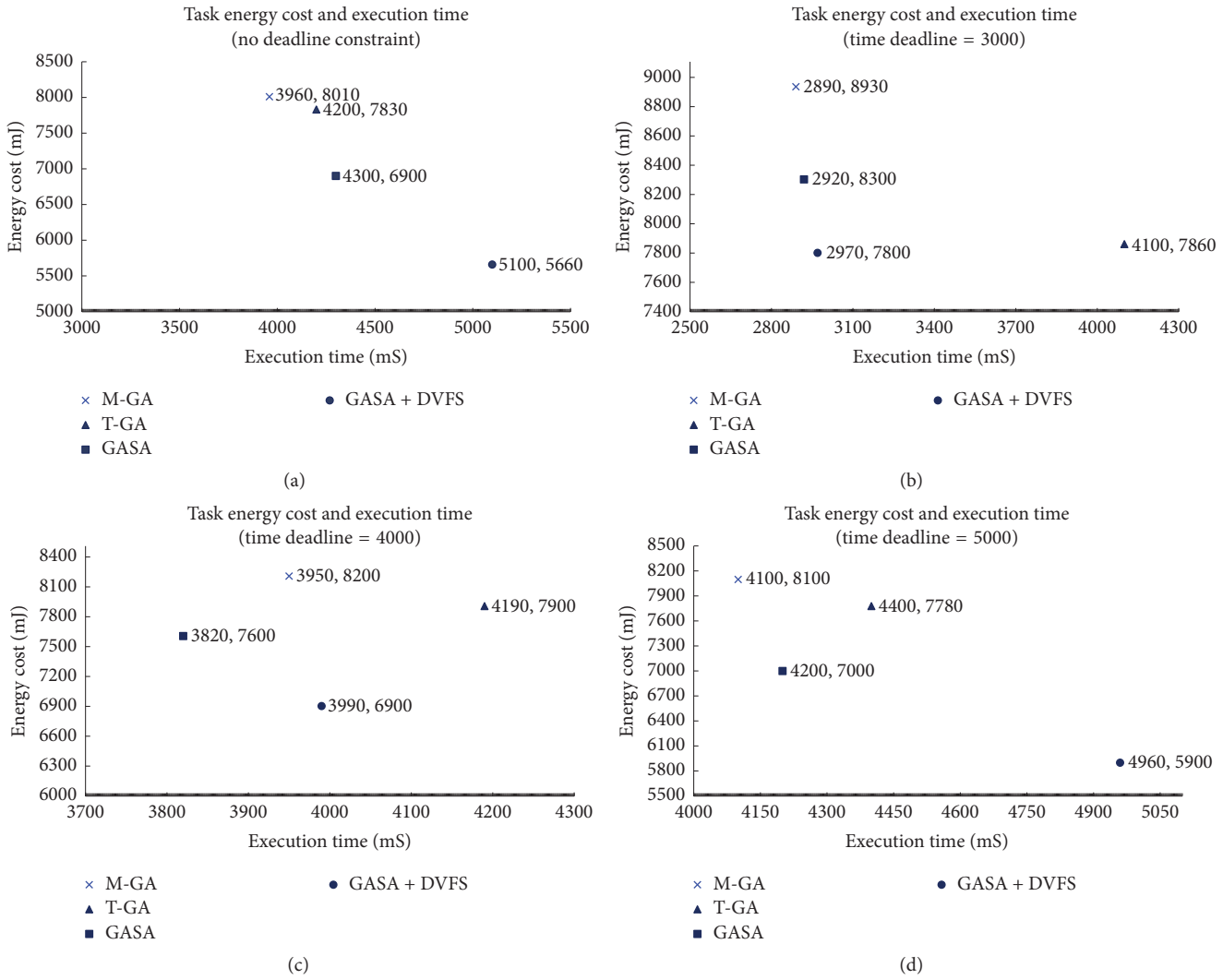

Figure 5: Energy cost with different time constraints. (a) No deadline constraint. (b) Deadline is set to 3000. (c) Deadline is set to 4000. (d) Deadline is set to 5000.

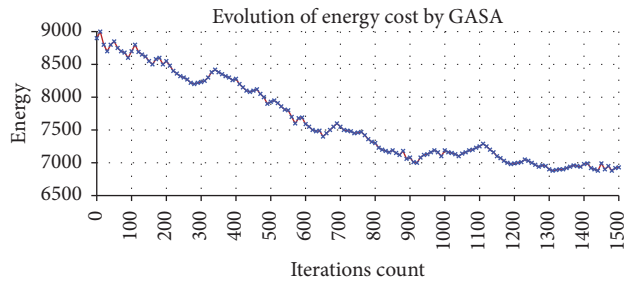

Figure 6: Energy evolution process when using the GASA algorithm.

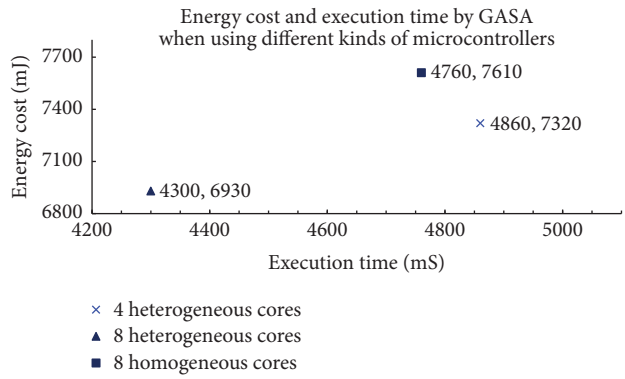

Figure 7: Energy cost and execution time on different kinds of microcontrollers.

captured data, and then transmits the data packet to the manager.

Several microcontrollers need to be equipped on the multicore node, and the selection of the microcontrollers depends on the software features. In the fire detection scene, two typical kinds of application tasks exist. One is the scalar-data task such as the temperature and gas sampling. The other

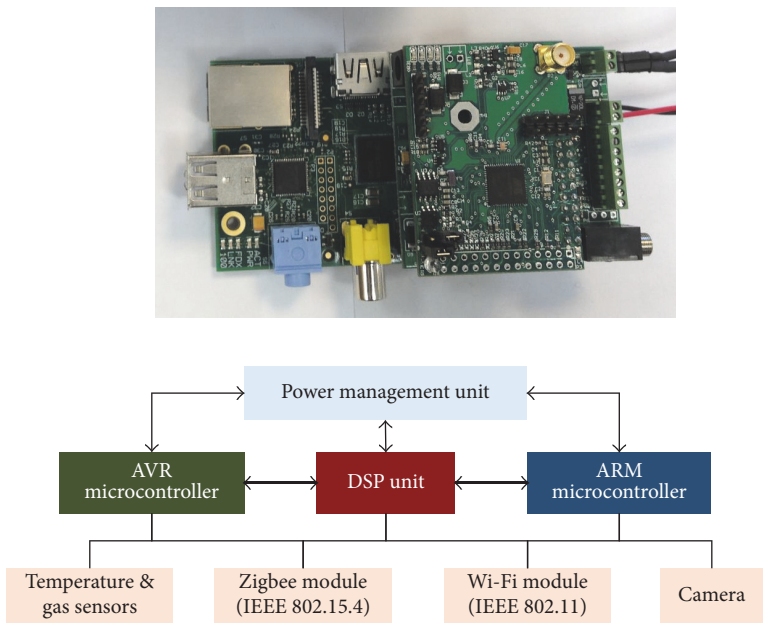

Figure 8: Prototype board and circuit diagram of the fire detection multicore WMSN node.

Table 2: Multicore scheduling result by using the GASA algorithm.

Fire detection application tasks	Current and time of running tasks on each core						Scheduling result by GASA
	AVR core		ARM core		DSP unit		
	I (mA)	t (ms)	I (mA)	t (ms)	I (mA)	t (ms)	
Scalar data sampling	14.5	580	17.6	550	N/A		AVR
Multimedia data sampling	69.7	1120	55.3	980			ARM
Data storage (1K bytes)	16.3	1030	20.9	580			ARM
Signal processing	9.9	2680	19.7	50	26.3	18	DSP
Scalar data transmission	20.9	132	21.8	139	N/A		AVR
Multimedia data transmission	N/A		375	32			ARM

is the multimedia task such as the image compression and signal processing. These two tasks have different features and are appropriate to run on different types of microcontrollers. The scalar-data task is simple and can be performed by the low-power general CPU, whereas the multimedia task is computation intensive and is appropriate to run on the specific units such as the DSP or GPU (Graph Processor Unit). By equipping more than one type of microcontroller on the node, the features of the WMSN hardware can better match that of the software. Consequently, the energy and delay performance of the WMSN node can be improved.

In terms of the discussion above, we design a fire detection WMSN node MiLive-v2 as is shown in Figure 8. Three primary microcontrollers are equipped on this node. One is the 8-bit low-power tiny microcontroller AVR ATmega128rfa1, which can be used to run the simple tasks. One is the 32-bit powerful ARM microcontroller ARM1176JZF, which can be used to run the more complicated tasks. The other is the DSP unit, which can be specially used to perform the image processing. These microcontrollers can communicate with each other through the GPIO ports, and any microcontroller can control the working status (sleep, active, or power off) of the others by sending the control commands. If a microcontroller has no tasks to run for the moment, it can enter the idle status to save the energy. If it has no tasks to run for a long time, it can be powered off directly. The BitCloud software firmware [34] is burned on the AVR microcontroller while the Linux firmware is burned on the ARM microcontroller.

7.2. Real-World Performance Evaluation. We applied the GASA algorithm to the multicore node MiLive-v2 and get the multicore task scheduling result as is illustrated in Table 2. There are six kinds of tasks in this application. They are, respectively, the temperature and gas sampling, the image and video sampling, data storage, signal processing, low-rate scalar-data transmission, and high-rate multimedia data transmission.

The current and time of running each task on different cores are measured, and the result is shown in Table 2. From

Table 3: Energy and delay cost of running fire detection application on different platforms.

Platform	Energy cost (mJ)	Execution time (S)	Deadline guarantee
Single-core AVR	127.9	4.1	No
Single-core ARM	89.7	2.43	Yes
Multicore MiLive-v2	63.6	2.36	Yes

Table 2, it is seen that each task is appropriate to run on the special cores. The signal processing task is efficient to run on the DSP unit rather than on the AVR core. This is because more multipliers are embedded inside the DSP circuit. The scalar-data sampling task is more appropriate to run on the AVR core. This is because the circuit construction of the AVR core is simpler and less energy cost will be required by its operations. The scalar-data packet is more energy efficient to be transmitted by the Zigbee module rather than the Wi-Fi module. This is because Zigbee is a specific low-cost, low-power, wireless mesh network standard dedicated to low-battery devices [35]. However, for the transmission of large amount of multimedia data, the Wi-Fi module will be more appropriate.

To evaluate the optimization performance of multicore MiLive-v2, we run the fire detection application, respectively, on the MiLive-v2, the single-core ARM platform, and the single-core AVR platform. The energy cost and execution time on these platforms are shown in Table 3. From the result, it is seen that the single-core AVR platform has the highest energy cost and longest execution time. This is because the AVR core is an 8-bit tiny microcontroller, and its hardware circuit is not appropriate to run the computation-intensive tasks. The single-core ARM platform has less energy cost and execution time. This is because the ARM core is 32 bits and can perform the computation-intensive tasks more efficiently. Suppose the deadline of the tasks is 2.43 seconds; then the energy cost of these platforms is, respectively, 63.6, 89.7, 127.9. That is, the energy cost of the multicore MiLive-v2 platform can be optimized by 50.3% and 29%, respectively, if compared to the single-core AVR platform and the single-core ARM platform.

8. Conclusions

This paper investigates an energy-efficient, delay-efficient, hardware and software cooptimization platform which aims to conserve the energy while guaranteeing the deadline of the real-time WMSN tasks. In the hardware aspect, an energy-efficient, delay-efficient, reconfigurable hardware platform is designed. This platform uses the heterogeneous multicore architecture and the DVFS adaption technique. With the heterogeneous multicore architecture, each WMSN task can be assigned to the most appropriate microcontroller to be executed. With the DVFS adaption technique, the hardware characteristic of each microcontroller can be adjusted dynamically in terms of the software run-time contexts. By these means, the WMSN hardware platform can become highly context aware of the software. Consequently, the software can be executed more efficiently with less energy cost and execution time. In the software aspect, an energy and delay multiobjective optimization algorithm GASA based on the hardware platform above is researched. GASA incorporates the genetic algorithm with the simulated annealing algorithm so that the global optimization solution can be searched out within acceptable time. In addition to the GASA, a DVFS adaption algorithm is also investigated. This algorithm adjusts the working frequency of each task intelligently so as to strip the time redundancy during the task executing process. By combining the DVFS with the GASA, the energy efficiency of the WMSN nodes can be improved significantly even under strict constraint of the execution time. To evaluate the optimization performance, we did both the simulation and the real-world experiment works. The simulation results prove the heterogeneous multicore architecture, and the DVFS adaption mechanisms are effective in optimizing the WMSN energy and delaying performance. Moreover, the GASA algorithm is convergent and can search out better optima than the traditional GA. The real-world experiment results show that the energy cost of the multicore WMSN node can be optimized by 50.3% and 29%, respectively, if compared to the traditional single-core AVR node and single-core ARM node.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

Xing Liu and Haiying Zhou make equal contributions to this article.

Acknowledgments

Our thanks go to the support from National Natural Science Foundation of China (Grants nos. 61702387, 61672398, 61771354), Provincial Science & Technology Pillar Program of Hubei (Grants nos. 2017AAA027, 2017AAA042), Provincial Science & Technology International Cooperation Program of Hubei (Grant no. 2017AHB048), Key Natural Science Foundation of Hubei Province of China (Grants nos. 2015CFA069, 2017CFA012), and Natural Science Foundation of Hubei Province of China (Grant no. 2017CFB302).

References

[1] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "Wireless multimedia sensor networks: applications and testbeds," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1588–1605, 2008.

[2] R. Hemalatha, S. Radha, and S. Sudharsan, "Energy-efficient image transmission in wireless multimedia sensor networks using block-based Compressive Sensing," *Computers and Electrical Engineering*, vol. 44, pp. 67–79, 2015.

[3] R. Banerjee, S. Chatterjee, and S. Das Bit, "An energy saving audio compression scheme for wireless multimedia sensor networks using spatio-temporal partial discrete wavelet transform," *Computers and Electrical Engineering*, vol. 48, pp. 389–404, 2015.

[4] T. Ma, M. Hempel, D. Peng, and H. Sharif, "A survey of energy-efficient compression and communication techniques for multimedia in resource constrained systems," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 963–972, 2013.

[5] Y. Sun, H. Luo, and S. K. Das, "A trust-based framework for fault-tolerant data aggregation in wireless multimedia sensor networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, pp. 785–797, 2012.

[6] H. R. Dhasian and P. Balasubramanian, "Survey of data aggregation techniques using soft computing in wireless sensor networks," *IET Information Security*, vol. 7, no. 4, pp. 336–342, 2013.

[7] F. Deniz, H. Bagci, I. Korpeoglu, and A. Yazici, "An adaptive, energy-aware and distributed fault-tolerant topology-control algorithm for heterogeneous wireless sensor networks," *Ad Hoc Networks*, vol. 44, pp. 104–117, 2016.

[8] A. A. Aziz, Y. A. Şekercioğlu, P. Fitzpatrick, and M. Ivanovich, "A survey on distributed topology control techniques for extending the lifetime of battery powered wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 121–144, 2013.

[9] H. Shen and G. Bai, "Routing in wireless multimedia sensor networks: a survey and challenges ahead," *Journal of Network and Computer Applications*, vol. 71, pp. 30–49, 2016.

[10] S. Ehsan and B. Hamdaoui, "A survey on energy-efficient routing techniques with QoS assurances for wireless multimedia sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 265–278, 2012.

[11] B. Yahya and J. Ben-Othman, "Towards a classification of energy aware MAC protocols for wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 9, no. 12, pp. 1572–1607, 2009.

[12] T. AlSkaif, B. Bellalta, M. G. Zapata, and J. M. Barcelo Ordinas, "Energy efficiency of MAC protocols in low data rate wireless multimedia sensor networks: a comparative study," *Ad Hoc Networks*, vol. 56, pp. 141–157, 2017.

[13] G. Han, Y. Dong, H. Guo, L. Shu, and D. Wu, "Cross-layer optimized routing in wireless sensor networks with duty cycle and energy harvesting," *Wireless Communications and Mobile Computing*, vol. 15, no. 16, pp. 1957–1981, 2015.

[14] Z. Hamid and F. B. Hussain, "QoS in wireless multimedia sensor networks: a layered and cross-layered approach," *Wireless Personal Communications*, vol. 75, no. 1, pp. 729–757, 2014.

[15] D. G. Costa and L. A. Guedes, "A survey on multimedia-based cross-layer optimization in visual sensor networks," *Sensors*, vol. 11, no. 5, pp. 5439–5468, 2011.

[16] B. Kanagal and A. Deshpande, "Online filtering, smoothing and probabilistic modeling of streaming data," in *Proceedings of the IEEE 24th International Conference on Data Engineering (ICDE '08)*, pp. 1160–1169, Washington, DC, USA, April 2008.

[17] I. Ha, M. Djuraev, and B. Ahn, "An energy-efficient data collection method for wireless multimedia sensor networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 9, Article ID 698452, 2014.

[18] C. Zhu, H. Zhang, G. Han, L. Shu, and J. J. P. C. Rodrigues, "BTDGS: binary-tree based data gathering scheme with mobile sink for wireless multimedia sensor networks," *Mobile Networks and Applications*, vol. 20, no. 5, pp. 604–622, 2015.

[19] A. Alanazi and K. Elleithy, "Real-time QoS routing protocols in wireless multimedia sensor networks: study and analysis," *Sensors*, vol. 15, no. 9, pp. 22209–22233, 2015.

[20] M. Doudou, D. Djenouri, N. Badache, and A. Bouabdallah, "Synchronous contention-based MAC protocols for delay-sensitive wireless sensor networks: a review and taxonomy," *Journal of Network and Computer Applications*, vol. 38, no. 1, pp. 172–184, 2014.

[21] M. Doudou, D. Djenouri, and N. Badache, "Survey on latency issues of asynchronous MAC protocols in delay-sensitive wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 528–550, 2013.

[22] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36)*, pp. 81–92, San Diego, Claif, USA, December 2003.

[23] X. Liu, K. M. Hou, C. de Vaulx et al., "Memory and energy optimization strategies for multithreaded operating system on the resource-constrained wireless sensor node," *Sensors*, vol. 15, no. 1, pp. 22–48, 2015.

[24] X. Liu, K. M. Hou, C. De Vaulx, K. El Gholami, and S. Xiong, "LiveWSN: a memory-efficient, energy-efficient, reprogrammable, and fault-tolerant platform for wireless sensor network," *International Journal of Distributed Sensor Networks*, vol. 12, no. 9, 2016.

[25] B. Brock and K. Rajamani, "Dynamic power management for embedded systems," in *Proceedings of the IEEE International SOC Conference [Systems-on-Chip]*, pp. 416–419, Portland, OR, USA, September 2003.

[26] W. Kim, M. S. Gupta, G. Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proceedings of the IEEE 14th International Symposium on High Performance Computer Architecture (HPCA '08)*, pp. 123–134, Salt Lake City, Utah, USA, February 2008.

[27] E. Horowitz and S. Sahni, "Exact and approximate algorithms for scheduling non-identical processors," *Journal of the ACM*, vol. 23, no. 2, pp. 317–327, 1976.

[28] M. R. Garey and D. S. Johnson, *Computers and Intractability*, vol. 29, W. H. Freeman and Company, New York, NY, USA, 2002.

[29] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, Massachusetts, USA, 1998.

[30] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[31] Y. Monnier, J. P. Beauvais, and A. M. Deplanche, "A genetic algorithm for scheduling tasks in a real-time distributed system," in *Proceedings of the 24th Euromicro Conference*, vol. 2, pp. 708–714, Vasteras, Sweden, August 1998.

[32] M. D. Theys, T. D. Braun, H. J. Siegal, A. A. Maciejewski, and Y. K. Kwok, "Mapping tasks onto distributed heterogeneous computing systems using a genetic algorithm approach," in *Solutions to Parallel and Distributed Computing Problems: Lessons from*

Biological Sciences, pp. 135–178, John Wiley and Sons, New York, NY, USA, 2001.

[33] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[34] BitCloud—ZigBee PRO, Atmel corporation, 2017, http://www.atmel.com/tools/bitcloud-zigbeepro.aspx.

[35] ZigBee Specification, ZigBee Alliance, 2017, http://www.zigbee.org.