Networked Architecture for Hybrid Electrical Energy Storage Systems [*]

Younghyun Kim, Sangyoung Park, and Naehyuck Chang
Seoul National University, Seoul, Korea
{yhkim, sypark, naehyuck}@elpl.snu.ac.kr

Qing Xie, Yanzhi Wang, and Massoud Pedram
University of Southern California, Los Angeles, CA, USA
{xqing, yanzhiwa, pedram}@usc.edu

ABSTRACT

A hybrid electrical energy storage (HEES) system that consists of multiple, heterogeneous electrical energy storage (EES) elements is a promising solution to achieve a cost-effective EES system because no storage element has ideal characteristics. The state-of-the-art HEES systems are based on a shared-bus charge transfer interconnect (CTI) architecture. Consequently, they are quite limited in scalability which is a function of the number of EES banks. This paper is the first introduction of a HEES system based on a networked CTI architecture, which is highly scalable and is capable of accommodating multiple, concurrent charge transfers. The paper starts by presenting a router architecture for the networked CTI and an effective on-line routing algorithm for multiple charge transfers. In the proposed algorithm, negotiated congestion (NC) routing for multiple charge transfers is performed and any lack of routing resources is addressed by merging two or more charge transfers while maximizing the overall energy efficiency by setting the optimal voltage level for the shared CTI. Examples of the proposed networked CTI are presented and the efficacy of the routing algorithm is demonstrated on a mesh-grid networked CTI.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

General Terms

Algorithms, Design

Keywords

Hybrid electrical energy storage, Charge transfer interconnect, Routing algorithm

[*]This work was supported by the Brain Korea 21 Project and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0016480). The ICT at Seoul National University provides research facilities for this study.

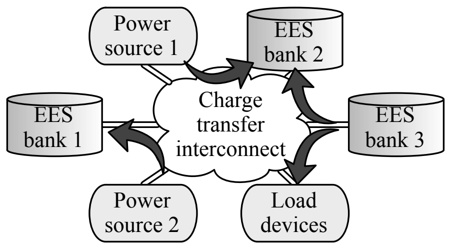


Figure 1: A hybrid electrical energy storage system with three EES banks.

1. INTRODUCTION

Hybrid electrical energy storage (HEES) systems consist of multiple, heterogeneous electrical energy storage (EES) elements [1, 2]. They utilize the strengths of each EES element type while hiding its disadvantages by performing appropriate charge management processes, including charge replacement, charge allocation, and charge migration [3, 4, 5]. HEES systems are one of the key resources in a smart grid enabling energy management, peak shaving, load balancing, and the like [6, 7]. They also improve the generation efficiency of renewable power sources such as solar cells and windmills by enabling maximum power extraction from these sources because of the ability to store the excess generated energy for future use, and by decoupling the load demand variation from the power generation [8].

Figure 1 shows a conceptual diagram of a HEES system composed of three EES banks. The EES banks can freely exchange energy among each other by transferring charge over the charge transfer interconnect (CTI). Power sources and load devices are also connected to the CTI. Previously reported HEES systems rely on a simple CTI architecture e.g., a single shared-bus CTI. However, a shared-bus CTI has a major limitation in terms of the HEES system scalability, that is, the number of EES banks is limited to a rather small count because of the potential contention on the CTI bus. Note that the CTI bus contention is different from contention in computer network in that the objects to be routed (charges) can be merged together into a single object. The contention of a CTI bus is thus defined in terms of the overall charge transfer efficiency. Each charge transfer task has its own optimal CTI voltage level to achieve the maximum transfer efficiency [3, 4, 5]. Merging the charge transfers is possible only at the expense of charge transfer efficiency degradation since transfer tasks have to share a CTI bus which can only have single (likely sub-optimal) CTI voltage. Severe efficiency degradation may thus offset the benefits of merging charge transfer tasks in the HEES system if the optimal CTI voltage levels of candidate charge transfer tasks are manifestly different.

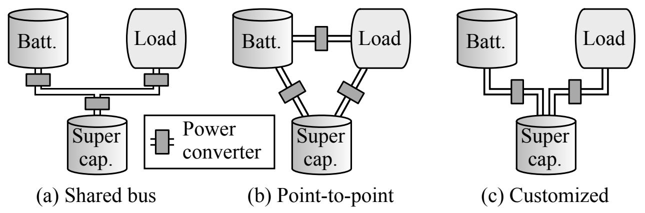


Figure 2: Various CTI architectures in view of the power path.

The solution to the above-mentioned problem is to use multiple CTIs in a HEES system. Multiple CTIs would then enable concurrent charge transfers with high energy efficiency by providing more routing resources and by relieving the single CTI voltage constraint. This paper is the first to introduce a networked CTI architecture for HEES systems to accommodate multiple EES banks and multiple, concurrent charge transfers while ensuring scalability. Since this is the first paper of a networked CTI for HEES systems, we consider the most basic mesh network topology.

This paper addresses the following important issues in charge transfers in networked CTI HEES systems. The first issue is routing of charge transfer paths in mesh networks as in traditional field-programmable gate array (FPGA) signal interconnects. We devise a CTI routing algorithm exploiting the similarity between our routing problem and the FPGA routing problem. We also discuss merging of charge transfer tasks and its impact on the charge transfer energy efficiency. Although the CTI routing in a networked architecture has similarities with the on-chip signal interconnects routing for an FPGA, there is a fundamental difference such that the charge transfers can share a routing path while on-chip signals cannot. We perform merging of charge transfers considering energy efficiency to eliminate routing failures due to lack of routing resources. The efficiency of a charge transfer task depends on the efficiency of converters, which are in turn affected by the CTI voltage and amount of charge to be transferred. We devise an algorithm to select charge transfer tasks to be merged based on their optimal CTI voltage values and congestions among tasks. We formulate the CTI routing problem and devise a systematic method to effectively perform energy-efficient charge transfers in networked CTI HEES systems.

The contributions of this paper are summarized as follows: i) we introduce a networked CTI architecture for HEES systems; ii) we define and precisely formulate the CTI routing problem on a networked CTI; and iii) we propose a computationally efficient online algorithm for the problem with the objective of maximizing the charge transfer efficiency. We demonstrate examples of the proposed networked CTI architecture and show the efficacy of the devised algorithm.

2. RELATED WORK

2.1 HEES CTI Architectures

Figure 2 shows three representative examples of the CTI architectures proposed in the previous HEES systems in view of the power paths. Shared-bus CTI architectures (typically called DC bus) are commonly used when the number of EES banks is limited. Recent works on the HEES system management methodologies [3, 4, 5, 9] assume a general shared-bus CTI architecture (Figure 2(a)). The shared-bus CTI is analogous to an on-chip shared bus on a system-on-chip (SoC) and their advantages and disadvantages are similar. Another architecture is a complete connection among the nodes [7] (Figure 2(b)). This architecture is comparable to a point-to-point connection in an SoC. Both the shared-bus and point-to-point connection architectures are feasible as long as the number of EES banks is small, but they certainly lack scalability to accommodate a large-scale HEES system. The other architecture is a customized network architecture for a particular application and operation policy. For example, a supercapacitor buffer efficiently mitigates the rate-capacity effect of a Li-ion battery especially for pulsed load demand [6] (Figure 2(c)). As the control policy is to use the supercapacitor as a buffer of the battery, the path from the battery bank to the load device is not necessary. This architecture is similar to a network-on-chip (NoC) architecture with irregular connectivity which is fully dependent on the application. In short, none of the previously introduced CTI architectures can be used to accommodate a large number of EES banks for general applications.

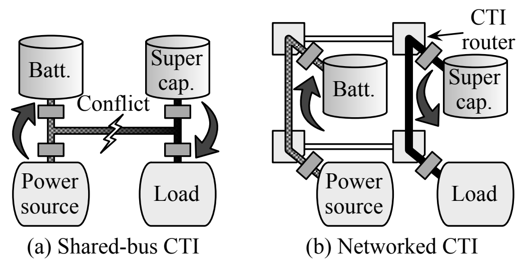


Figure 3: Shared-bus CTI and networked CTI of four nodes.

2.2 Conventional Routing Problems

The CTI routing problem in a networked CTI has similarity to the conventional FPGA signal routing problem. In the problem of CTI routing, each task competes for routing resources such as converters and CTI links, whereas each signal competes for wires and connection points in FPGA routing. The FPGA routing is a highly complex combinatorial optimization problem, and thus it is usually done by iterative rip-up and reroute of signals. The success of routing is dependent not just on the choice of which nets to reroute, but also on the order in which rerouting is done as shown in traditional rip-up and reroute methods [10, 11]. The negotiation-based FPGA router successfully relieves the signal ordering problem and provides a systematic rip-up and reroute capability [12]. This routing algorithm allows initial sharing of the routing resources among signals, but subsequently makes them negotiate for the shared resource with other signals until no resource is shared. The negotiation-based routing algorithm is further enhanced in terms of compilation time by incorporating delay-driven routing [13]. More recent works such as [14] focus on the new architecture or technology scaling, but the core of the routing algorithm is still based on [12].

3. CHARGE TRANSFER INTERCONNECT

3.1 Charge Transfer Conflicts

The charge management of a HEES system is achieved by charge allocation, replacement, and migration operations [1]. The operations are basically charge transfers among EES banks using the CTI as charge transfer medium. The previous works on the charge management of HEES [3, 4, 5] assumed that the charge transfer path is always available for a given charge transfer task. They focused on maximizing the energy efficiency by setting a proper value for CTI voltage of the charge transfers. However, it is not always true that a charge transfer path is available whenever it is required. Two or more charge transfer tasks can have a conflict by competing for the shared-bus. Figure 3(a) demonstrates an example where the

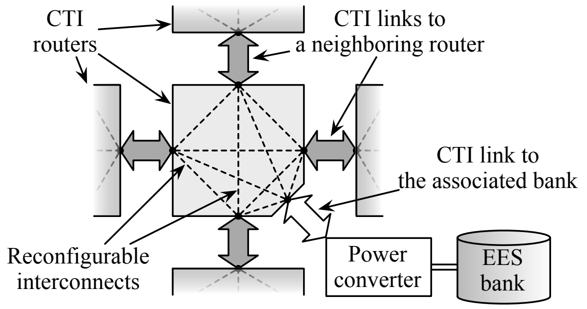

CTI routers
CTI links to a neighboring router
CTI link to the associated bank
Reconfigurable interconnects
Power converter
EES bank

Figure 4: Architecture of a CTI router. An associated EES bank is connected via a power converter. The arrows denote the CTI links.

power supply charges the battery bank and the supercapacitor bank supplies power to the load at the same time. Two charge transfer tasks have different optimal CTI voltage values, which maximize the charge transfer energy efficiency of each task, and there is only one CTI link.

We define that two or more charge transfers *conflict* when they try to occupy the same CTI link and have different optimal CTI voltage values. Such a conflict enforces the charge transfer tasks to use the same CTI voltage, and thus at least one of them has to suffer possibly severe degradation in energy efficiency. Separating the two charge transfers in time domain by scheduling may mitigate the conflict, but we leave this as a future work.

3.2 Networked CTI Architecture

We introduce a *networked CTI architecture* as shown in Figure 3(b) to fundamentally solve the charge transfer conflict problem, which ensures scalability to a large number of EES banks. Specifically, we use a mesh interconnect architecture to ensure flexibility and scalability of networked CTI architecture. One important component to realize the networked CTI architecture is the CTI router. We propose a CTI router that connects CTI links, an associated component (i.e., an EES bank, a power source, or a load device), and a power converter. Figure 4 shows the detailed architecture of the CTI router. Each CTI router is connected with the adjacent CTI routers through the CTI links. The CTI router consists of reconfigurable interconnects which are denoted as dashed lines in Figure 4. We dynamically connect or disconnect the reconfigurable interconnects inside the router to setup a path from one CTI link to another. The reconfigurable interconnects form a complete graph so that the signal can be routed in any direction. The CTI router in Figure 4 has five CTI links, and thus it has ten interconnects each of which is implemented as a pair of back-to-back MOSFET switches. We adopt the switching power converter efficiency model from [15] in this paper.

The networked CTI architecture is comparable to a general NoC architecture. As the number of processing elements in an SoC increases, the single-level on-chip bus architecture is no longer able to handle increased data exchanges between the processing elements. Similar to the NoC which requires packet routing, a HEES system with a networked CTI architecture requires routing of the charge transfers. However, CTI routing on a networked CTI is not the same as the conventional NoC packet routing, conventional signal routing for FPGA, nor application-specific integrated circuit (ASIC). To efficiently describe the networked CTI architecture routing problem, we compare it with the conventional signal routing problem as shown in Table 1.

Table 1: CTI routing and signal routing problem mapping.

	Networked CTI routing	Signal routing
Nodes	EES banks, power sources, load devices	Processing elements
Links	CTI links	On-chip interconnects
Flows	Charge flows	Signal flows
Objective	High efficiency	Low latency
Output	Charge routing trees w/ voltage and current	Signal routing trees w/ buffer size
Resource sharing	Allowed	Not allowed
Routability	Guaranteed w/ resource sharing	Not guaranteed

4. PROBLEM STATEMENT

4.1 Formal Definitions

We present a formal definition of the CTI routing problem in this section. We first define a *node* as a combination of a CTI router and either of an EES bank, a power source, or a load device associated with it. A CTI network is a graph $G = (V, E)$ where V is a set of vertices that corresponds to nodes, and E is a set of edges that corresponds to CTI links between two elements in V. It is an undirected graph as the CTI links are bidirectional electrical conductors. The link between the CTI router and the associated EES element (an EES bank, a power source or a load device) is a dedicated resource, and thus we do not consider this in the routing algorithm.

We define a set of charge transfer tasks such that $\tau = \{T_1, T_2, \ldots, T_k\}$, where k is the number of tasks. Each charge transfer task is a two-tuple such as $T_i = (\sigma_i, \delta_i)$, where σ_i is the set of source nodes and δ_i is the set of destination nodes. The task describes the nodes that should be connected by the routing algorithm. The deadline of the transfer and the amount of charge provided by the source nodes or received by destination nodes are defined separately because they are not related to the routing process. They are used for charge transfer optimization discussed in Section 4.3. We add a newly arriving task to τ or remove a finished task from τ, update remaining time until the deadline of the existing or remaining tasks, and perform routing again.

The CTI routing problem is to find routing paths for a given transfer task set τ, that connects all the nodes in σ_i and δ_i for each $T_i \in \tau$. A node of T_i participates in only one charge transfer, and it is either a source or a destination, not both. That is,

$$\bigcup_{T_i \in \tau} (\sigma_i \cap \delta_i) = \varnothing \quad \text{and} \quad \bigcap_{T_i \in \tau} (\sigma_i \cup \delta_i) = \varnothing. \tag{1}$$

As a result of the CTI routing, a disjoint subset of edges in E that forms an acyclic routing tree is assigned to each T_i. We set each CTI router configuration (make connections of the internal interconnects) according to the edges in the routing trees. An individual routed charge transfer is equivalent to a charge transfer on an independent shared-bus CTI. Therefore, it enables us to apply any previous HEES charge management methods that are based on a shared-bus CTI to each routed charge transfer task.

4.2 Charge Transfer Interconnect Routing

The CTI routing problem resembles to the FPGA routing problem discussed in Section 2.2 in the aspect that the routing resources are discrete and scarce. The routing process allocates limited resources to the nets (the set of charge transfer tasks or signals), and each net is allowed to use the resource for a designated period.

Routing charge transfer tasks requires iterative execution of two steps; i) the CTI routing and ii) charge transfer optimization. The CTI routing operation is to determine a routing path of the charge

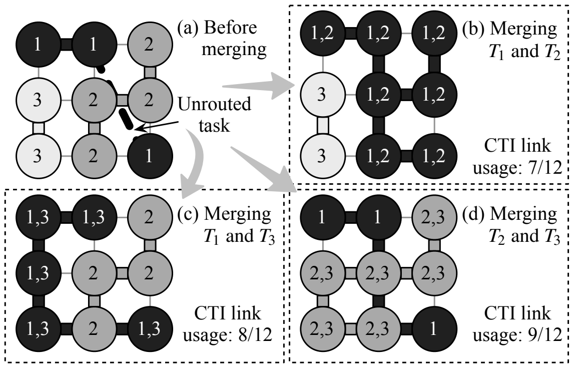


Figure 5: Example routing of three tasks after merging. T_1 is unrouted in (a), and routing after three possible merging combinations are presented in (b), (c), and (d).

transfer, and the charge transfer optimization operation is to determine the voltage level of the routing path and the amount of current through the routing path. We discuss the routing and optimization one after another in this section and in Section 4.3, respectively.

The CTI routing problem should tackle limitation in the routing resources (the CTI links) like the conventional FPGA routing problems. Signal routing of FPGA fails if there are unrouted nets which are not routable with remaining routing resources. The workaround is either increasing the resource, i.e., using a larger device or optimizing placement so that the congestion is reduced.

On the other hand, redoing placement is not an option for the CTI routing problem because the nodes are at a fixed location in the HEES system and cannot be moved. Instead, we perform *merging* in order to mitigate the routing congestion. This is a unique feature of the CTI routing for HEES systems. Merging is combining two charge transfer tasks into one to produce a new task set. Two or more migration tasks can be merged and share resources unlike signal routing.

If one task has a longer deadline than the other, the combined task uses the CTI links for whichever the shorter deadline. After the deadline expires, the task with a shorter deadline releases the CTI links and the task with a longer deadline solely occupies the CTI links after rerouting. Merging $T_i = (\sigma_i, \delta_i)$ and $T_j = (\sigma_j, \delta_j)$ results in a new task $T_{i,j} = (\sigma_i \cup \sigma_j, \delta_i \cup \delta_j)$. Then T_i and T_j are removed from τ and $T_{i,j}$ is added to τ. After T_i or T_j that has a shorter deadline is finished, the remaining task is added back to τ with the remaining deadline.

Figure 5 is an example of the merging to improve routability of three tasks. In Figure 5(a), T_2 and T_3 are routed, but T_1 is not routed. There are three possible combinations to merge two tasks out of three as shown in Figures 5(b), (c), and (d). The CTI link usage out of 12 CTI links is different depending on the combinations. The number of unused CTI links directly affects the routability of the other charge transfer tasks.

Most importantly, merging is not free. A merged task suffers efficiency degradation due to single CTI voltage constraint. Therefore, we have to consider not only the routabiltiy but also the efficiency at same time. We discuss this matter in the Section 4.3.

4.3 Charge Transfer Optimization

The energy efficiency of a charge transfer is defined as ratio between the sum of the energy in all the EES banks after and before the transfer. Recent works on EES systems show that the energy efficiency of charge transfers is significantly affected by the CTI voltage and amount of transfer current from/to participating nodes [3, 4, 5]. The goal of charge transfer optimization is to mitigate the energy loss of charge transfer tasks by finding the best-suited CTI voltage and current. The optimization process considers the efficiency variation of the power converters according to the CTI voltage as well as electrical characteristics of all the EES elements. The input of the optimization process is a merged task, and the results are the CTI voltage, current from/to the nodes in σ and δ, and the overall energy efficiency.

When two or more tasks are merged into one, they have to share the CTI links that has a single voltage level. In case a task had substantially different CTI voltage from the CTI voltage after merging, the charge transfer efficiency would be greatly deteriorated. We choose the best task pairs to merge based on the optimal CTI voltage level similarity to avoid severe transfer efficiency degradation.

The power converter efficiency model and electrical characteristics of the EES elements are complex nonlinear functions that make the optimization process difficult. We adopt the derivation of the optimal CTI voltage and transfer current similar to the methods in the previous work [3, 4, 5] such as fractional optimization for efficiency and ternary search for the optimal CTI voltage. We do not discuss the detailed implementation of the optimization process because it is out of focus of this paper.

5. SOLUTION METHOD

5.1 CTI Link Cost Evaluation

We present the proposed networked CTI routing algorithm in Algorithm 1. The input of Algorithm 1 is the CTI network G and a set of charge transfer tasks τ. Algorithm 1 iteratively performs rip-up and rerouting the charge transfer tasks until all the tasks are routed. The kernel of the routing algorithm is based on the negotiated congestion (NC) routing algorithm in [12]. The cost of resources (CTI links) gradually increases over iterations, and each charge transfer task competes with others to occupy the resource. Only one charge transfer task that is willing to pay the cost occupies the resource, and the other tasks detour via other less-costly resources.

We first define the cost of resources taking into account the distinctive characteristics of the CTI routing problem. An edge $e = (u, v)$ is associated with a congestion cost $c[e]$ that is defined as

$$c[e] = (b[e] + h[e]) \cdot p[e], \tag{2}$$

where $b[e]$ is the base cost of the edge e, $h[e]$ is the congestion history cost and $p[e]$ is the penalty due to the congestion at the current iteration. The base cost $b[e]$ is related with the unit cost of charge transfer from u to v, and we set the base cost to 1. The penalty $p[e]$ is defined as

$$p[e] = 1 + p_{gradient} \cdot u[e], \tag{3}$$

where $p_{gradient}$ is a constant, and $u[e]$ is the number of charge transfer tasks that share the edge e. The congestion history cost $h[e]$ increases gradually after each iteration to increase cost of congested edge and make the conflicting nets to avoid it. That is,

$$h[e] = \begin{cases} h[e]' & \text{if } u[e] = 0 \\ h[e]' + h_{gradient} \cdot (u[e] - 1) & \text{if } u[e] \geq 1 \end{cases}, \tag{4}$$

where $h[e]'$ is $h[e]$ of the previous iteration, $h_{gradient}$ is a constant, and $h[e]$ is initially 0.

Only the congestion history cost is dependent on the number of iterations by (4), and it is a non-decreasing function of the number of iterations. This is because the nets to be routed do not change over iterations in the signal routing, and so the congested resources are likely to be congested again in subsequent iterations. This is not the case for the CTI routing problem because we merge conflicting tasks into one, and then the shared resources are not congested any more. The cost of the previously shared edges are overestimated if we do not decrease $h[e]$ after they are merged. This leads to other charge transfers to avoid using the released edges and results in non-optimal routing results. Therefore, we reduce $h[e]$ of edges that have been congested by the merged tasks.

5.2 Conflict Graph

We define a *conflict graph* as $G^c = (V^c, E^c)$. There are $k = |\tau|$ nodes in $V^c = \{v_1^c, v_2^c, \dots, v_k^c\}$, and each v_i^c is mapped to T_i. A conflict graph G^c is a complete graph, and each edge $e^c = (v_i^c, v_j^c) \in E^c$ is assigned with $d[e^c]$ which is *conflict count* between tasks T_i and T_j. Initially, $d[e^c]$ is set to zero, and we increase $d[e^c]$ by n if the tasks T_i and T_j share n CTI links. We define the sum of the conflict counts of all the edges in a conflict graph G^c to be a *conflict degree* $D[G^c]$ such that

$$D[G^c] = \sum_{e^c \in E^c} d[e^c], \tag{5}$$

which is the metric of routability of a given task set.

We also use this conflict graph to prune away the task pairs that do not increase the routability after merging. This is important to efficiently find task pairs to merge by avoiding a situation of trying all the pairs in every iteration. We try merging a pair of tasks, and accept it if it increases the routability. We define that the routability is improved if the conflict degree is reduced after merging by the conflict count between merged transfer tasks or more. That is, we accept the merging of T_i and T_j if

$$D[G^{c\prime}] \leq D[G^c] - d[(v_i^c, v_j^c)], \tag{6}$$

or reject it otherwise. We mark an edge of rejected task pair with $r[e^c] = 1$ to indicate the task pair is previously rejected, and $r[e^c] = 0$ otherwise.

We merge a pair of tasks T_i and T_j that have the least difference in the optimal CTI voltage if they conflict ($d[(v_i^c, v_j^c)] > 0$) and have not been rejected previously ($r[(v_i^c, v_j^c)] = 0$). Merging the two tasks results in a new conflict graph because two tasks T_i and T_j is removed and a new task $T_{i,j}$ is added. The new task is marked not-to-be-merged ($r[e^c] = 1$) with existing tasks if both the merged tasks were marked not-to-be-merged with the tasks. The conflict count $d[e^c]$ is reset to zero after merging. (Refer to Appendix A.1 for an example of the conflict graph management.)

5.3 Algorithm Description

The algorithm starts from initialization of the cost of $e \in E$ based on (2), (3), and (4) in Line 1. Initially, $u[e] = 0$ for all e. It also initializes the conflict graph G^c in Line 2. We try routing and merging until all the CTI links are not shared by multiple charge transfer tasks in the loop through Lines 3–15. The loop in Lines 4–6 attempts to route the given task set with the NC-router. The NC-router repeats rip-up and rerouting for all the charge transfer tasks while updating the edge cost $c[e]$ in Line 5. We update the conflict graph after one trial for the rip-up and rerouting for all the charge transfer tasks in Line 6. These procedures are repeated until the current task set τ is fully routed. The algorithm is terminated and returns the routing results after the charge transfer optimization for each task in Line 7 if the routing is successful.

We perform merging through Lines 8–15 if the routing fails. We judge that the task set is not routable if routing attempt fails for a certain number of iterations or a certain amount of runtime. The previous merging is rejected in Line 10 if it fails to improve the routability. If the previous merging is rejected, we restore the previous states of τ, G^c, and edge costs of E. We mark rejected pairs of tasks at the edges ($r[e^c] = 1$) in Line 11 so that they are not explored in the future attempts for merging.

Algorithm 1: Networked CTI routing algorithm

```
   Input: CTI graph $G$, Charge transfer task set $\tau$
   Output: Routing tree for each task with the optimal voltage
1  Initialize cost $c$
2  Initialize conflict graph $G^c$
3  while shared resource exists do
4      while routing retry conditions hold do
5          NC-route $\tau$ on $G_{cti}$ with cost $c$
6          Update conflict graph $G^c$
7      Solve the charge transfer optimization problem for each
       task
8      if routing failed then
9          if previous merging is not successful then
10             Reject the previous merging and restore $\tau$, $G^c$, and
               costs of $E$
11             Mark rejected pair of tasks in $G^c$
12         Save the current $\tau$, $G^c$, and costs of $E$
13         Merge the two tasks and update $\tau$
14         Update conflict graph $G^c$
15         Update costs of $E$
```

Merging tasks begins with saving the current states of τ, G^c, and edge costs of E so that we can restore them when the merging is rejected in Line 12. We utilize the conflict graph G^c to find candidate tasks to be merged as described in Section 5.2. We update G^c and reset the conflict count $d[e^c]$ to zero after merging in Line 14. We also update the cost c of CTI links based on the new CTI link utilization after merging in Line 15.

6. EXPERIMENTS

6.1 Experimental Setup

We demonstrate examples of the proposed networked CTI architecture and evaluate the proposed CTI routing algorithm compared with the state-of-the-art shared-bus CTI architecture in this section. The proposed CTI routing algorithm is not restricted to a specific topology, but we assume a CTI network of a regular-shape mesh-grid for the demonstration purpose. All the EES banks are supercapacitor banks, and thus the terminal voltage of each bank is linearly proportional to the state of charge and is initially different to each other. The initial terminal voltage of the EES banks is randomly determined between 15 V and 200 V.

The performance metric to be evaluated is the energy efficiency of charge transfer tasks. The baseline method is the shared-bus CTI architecture. We first begin with the charge transfers tasks that are single-source-single-destination (SSSD) ($|\sigma_i| = 1$ and $|\delta_i| = 1$ for all T_i). The SSSD transfers become multiple-source-multiple-destination (MSMD) transfers after merging. We do not lose any generality by assuming SSSD transfer tasks because the proposed algorithm can handle arbitrary number of nodes in σ and δ.

We assume the followings in charge transfers in the experiments. i) An SSSD transfer task defines the amount of energy into the destination node. The amount of charge transfer is defined from the destination side in an SSSD transfer task. The amount of energy from the source node is determined accordingly by the power converter efficiency. ii) The amount of energy into the destination nodes is kept the same in an MSMD transfer task after merging. We keep the ratio of the amount of energy to be discharged from each source the same.

Table 2: Routing results and efficiency of charge transfers in networked CTIs. All the tasks are initially SSSD transfers.

No.	Network grid size	Number of participating nodes	Number of tasks change	Energy efficiency	
				Networked CTI archi.	Shared-bus CTI archi.
1	3-by-3	4 out of 9	$2 \rightarrow 2$	88.5%	76.1%
2	3-by-3	8 out of 9	$4 \rightarrow 2$	79.2%	73.4%
3	5-by-5	12 out of 25	$6 \rightarrow 6$	81.2%	74.2%
4	5-by-5	18 out of 25	$9 \rightarrow 6$	57.7%	57.3%
5	7-by-7	18 out of 49	$9 \rightarrow 8$	81.6%	74.8%
6	7-by-7	38 out of 49	$19 \rightarrow 15$	75.9%	68.7%

6.2 Experimental Results

We use 3-by-3 to 7-by-7 mesh-grid CTI networks with different number of initial charge transfer tasks as benchmarks. Table 2 shows the number of nodes that participate in the charge transfer, total number of nodes, number of tasks in the initial and final output task sets, and the energy efficiency improvement compared with the shared-bus CTI architecture.

Figure 6 shows the input and output of the proposed algorithm with the benchmark No. 4 having a 5-by-5 CTI network and an initial task set of nine SSSD tasks. The CTI algorithm performs five times of routing and four times of merging (the last routing is not followed by merging). Three merges are accepted and one is rejected, and so the initial nine tasks are merged into six tasks as a result of the routing. Figure 6(b) shows that tasks T_1, T_6, and T_9 in Figure 6(a) are merged into T_1, and tasks T_4 and T_5 are merged into T_4. (Refer to Appendix A.2 for the detailed steps of routing.)

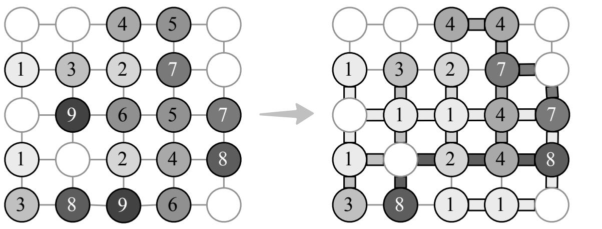
The experimental results show that the proposed routing algorithm successfully routes the charge transfer tasks even the number of tasks is large and the routing resources are limited. For example, there are initially 19 SSSD tasks in the benchmark No. 6, and so 38 nodes out of total 49 nodes participate in the charge transfers, which results in a very congested CTI network routing. The proposed algorithm merges only 4 of 19 tasks into other tasks and achieves a 7.2% higher energy efficiency compared with the charge transfers on a shared-bus CTI architecture. The energy efficiency improvement is up to 12.4% for the example benchmark set.

It is shown that the efficiency improvement diminishes as more number of tasks are merged. Benchmarks No. 1 and 2 end up with two tasks in the end, but efficiency improvement is less significant in No. 2 because we merge more number of tasks. It is the same for the benchmarks No. 3 and 4 that both end up with six tasks after merging. This is because more participating nodes in the same number of tasks imply that there are more nodes that do not have the optimal CTI voltage.

The energy efficiency improvement is significant when we consider the initial voltages of the EES banks are totally randomly generated. In fact, the energy efficiency improvement may be minor as in the benchmark No. 4 if the initial SSSD transfer tasks have a large voltage difference between the source and destination nodes. However, the benefit of the networked CTI architecture is larger when the voltage difference between the source and destination nodes is small in the initial charge transfer tasks before merging.

7. CONCLUSIONS

This paper introduced a networked charge transfer interconnect (CTI) architecture for hybrid electrical energy storage (HEES) systems. The networked CTI architecture is capable of accommodating an increased number of EES banks with an excellent scalability. We also described a CTI router, which is the basic building block of the proposed networked CTI architecture. Since HEES systems with networked CTI architecture require sophisticated control mechanisms, we presented a novel CTI routing algorithm that guarantees routability (despite routing congestion due to limited routing resources) by developing the charge transfer merging technique. The proposed CTI routing algorithm finds the routing paths for charge transfers, performing energy-efficient merging of tasks not to significantly degrade the overall energy efficiency. We showed that the proposed networked CTI architecture significantly improves the energy efficiency of the charge transfers by enabling multiple, concurrent transfers with optimal CTI voltages in comparison with the state-of-the-art shared-bus CTI architecture that all the charge transfers should be merged on the bus. The experimental results show that the proposed networked CTI architecture achieves up to 12.4% of energy efficiency improvement.



(a) Input CTI network and task set
(b) Output routing tree and task set

Figure 6: An example of routing result of nine tasks on a 5-by-5 regular-shaped mesh-grid CTI network.

8. REFERENCES

[1] M. Pedram, N. Chang, Y. Kim, and Y. Wang, "Hybrid electrical energy storage systems," in *ISLPED*, 2010, pp. 363–368.

[2] F. Koushanfar, "Hierarchical hybrid power supply networks," in *DAC*, 2010, pp. 629–630.

[3] Q. Xie, Y. Wang, Y. Kim, N. Chang, and M. Pedram, "Charge allocation for hybrid electrical energy storage systems," in *CODES+ISSS*, 2011, pp. 277–284.

[4] Q. Xie, Y. Wang, M. Pedram, Y. Kim, D. Shin, and N. Chang, "Charge replacement in hybrid electrical energy storage systems," in *ASP-DAC*, 2012, pp. 627–632.

[5] Y. Wang, Y. Kim, Q. Xie, N. Chang, and M. Pedram, "Charge migration efficiency optimization in hybrid electrical energy storage (HEES) systems," in *ISLPED*, 2011, pp. 103–108.

[6] D. Shin, Y. Kim, J. Seo, N. Chang, Y. Wang, and M. Pedram, "Battery-supercapacitor hybrid system for high-rate pulsed load applications," in *DATE*, 2011, pp. 875–878.

[7] A. Mirhoseini and F. Koushanfar, "HypoEnergy. hybrid supercapacitor-battery power-supply optimization for energy efficiency," in *DATE*, 2011, pp. 887–890.

[8] Y. Kim, Y. Wang, N. Chang, and M. Pedram, "Maximum power transfer tracking for a photovoltaic-supercapacitor energy system," in *ISLPED*, 2010, pp. 307–312.

[9] Y. Kim, S. Park, Y. Wang, Q. Xie, N. Chang, M. Poncino, and M. Pedram, "General balanced reconfiguration architecture for electrical energy storage banks," in *ICCAD*, 2011, pp. 624–631.

[10] D. Hill, "A CAD system for the design of field programmable gate arrays," in *DAC*, 1991, pp. 187–192.

[11] Y.-W. Chang, S. Thakur, K. Zhu, and D. F. Wong, "A new global routing algorithm for FPGAs," in *ICCAD*, 1994, pp. 356–361.

[12] L. McMurchie and C. Ebeling, "PathFinder: a negotiation-based performance-driven router for FPGAs," in *FPGA*, 1995, pp. 111–117.

[13] J. S. Swartz, V. Betz, and J. Rose, "A fast routability-driven router for FPGAs," in *FPGA*, 1998, pp. 140–149.

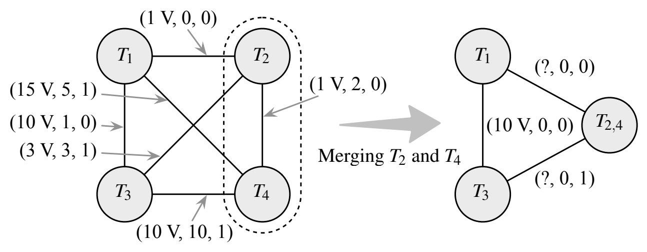
[14] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. M. Fang, and J. Rose, "VPR 5.0: FPGA cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *FPGA*, 2009, pp. 133–142.

[15] Y. Choi, N. Chang, and T. Kim, "DC–DC converter-aware power management for low-power embedded systems," *IEEE T. on CAD*, pp. 1367–1381, 2007.

APPENDIX

A.1 Conflict Graph Management Example

We give an example of task selection of tasks to be merged using a conflict graph and conflict graph update discussed in Section 5.3. Figure A1 shows an example of conflict graph update with four tasks. Figure A1(a) is the initial conflict graph G^c, and Figure A1(b) is the updated conflict graph $G^{c\prime}$ after merging. Each node corresponds to a charge transfer task. The three-tuple on each edge is $(\Delta V^{cti}_{opt}, d[e^c], r[e^c])$, where ΔV^{cti}_{opt} is the difference of the optimal CTI voltages of two tasks, $d[e^c]$ is the conflict count between two tasks, and $r[e^c]$ is the marking whether if the corresponding merging is rejected previously.



(a) G^c before merging (b) $G^{c\prime}$ after merging

Figure A1: Example of a conflict graph (a) before and (b) after merging tasks T_2 and T_4. Each edge is annotated with a three-tuple $(\Delta V^{cti}_{opt}, d[e^c], r[e^c])$.

We find a pair of tasks to merge from G^c as follows:

- We do not consider the task pairs for merging if they had been rejected before ($r[e^c] = 1$). This condition excludes the task pairs (T_1, T_4), (T_2, T_3) and (T_3, T_4).
- We do not consider the task pairs for merging if they are not conflicting ($d[e^c] = 0$). This condition excludes the task pair (T_1, T_2).
- Remaining task pairs are (T_1, T_3) and (T_2, T_4).
- We pick (T_2, T_4) for merging because the CTI voltage difference is 1 V, which is smaller than 10 V of (T_1, T_3).

We update the conflict graph to $G^{c\prime}$ after the merging as follows:

- T_2 and T_4 are removed and $T_{2,4}$ is added.
- All the conflict count $d[e^c]$ is reset to zero.
- Task pair (T_1, T_3) is not affected by the merging, and so $\Delta V^{cti}_{opt} =$ 10 V and $r[(v^c_1, v^c_3)] = 0$ for this pair remain the same.
- We increase $d[e^c]$ during the routing in Line 5 in Algorithm 1.
- ΔV^{cti} for (T_1, $T_{2,4}$) and (T_3, $T_{2,4}$) is set to unknown because the optimal CTI voltage for of $T_{2,4}$ is not calculated yet. It is calculated in Line 7 in Algorithm 1 after routing.
- We mark $T_{2,4}$ merge-able with T_1, i.e., $r[(v^c_1, v^c_{2,4})] = 0$ because T_2 was merge-able before merging. On the other hand, we mark $T_{2,4}$ not-to-be-merged with T_3, i.e., $r[(v^c_3, v^c_{2,4})] =$ 1 because neither T_2 nor T_4 was merge-able with T_3 before merging.

Since $D[G^c] = 21$ and $d[(v^c_2, v^c_4)] = 2$, merging T_2 and T_4 is accepted if $D[G^{c\prime}] \leq 21 - 2 = 19$ by (6), and the new conflict graph in Figure A1(b) is used in the next iteration. Otherwise (if rejected), we restore G^c, mark $r[(v^c_2, v^c_4)]$ to 1, and try merging the last remaining pair (T_1, T_3).

A.2 Routing Example

We show the detailed steps of CTI routing of Figure 6. The initial charge transfer task set has nine SSSD tasks. The steps illustrated in Figure A2 explain the iterations of the loop through Lines 3–15 of Algorithm 1 until the routing finishes. Figure A2(a) is the initial state, and it goes through five iterations as shown in Figures A2(b)–(f). The first merge is rejected and the following three merges are accepted, resulting in six tasks.

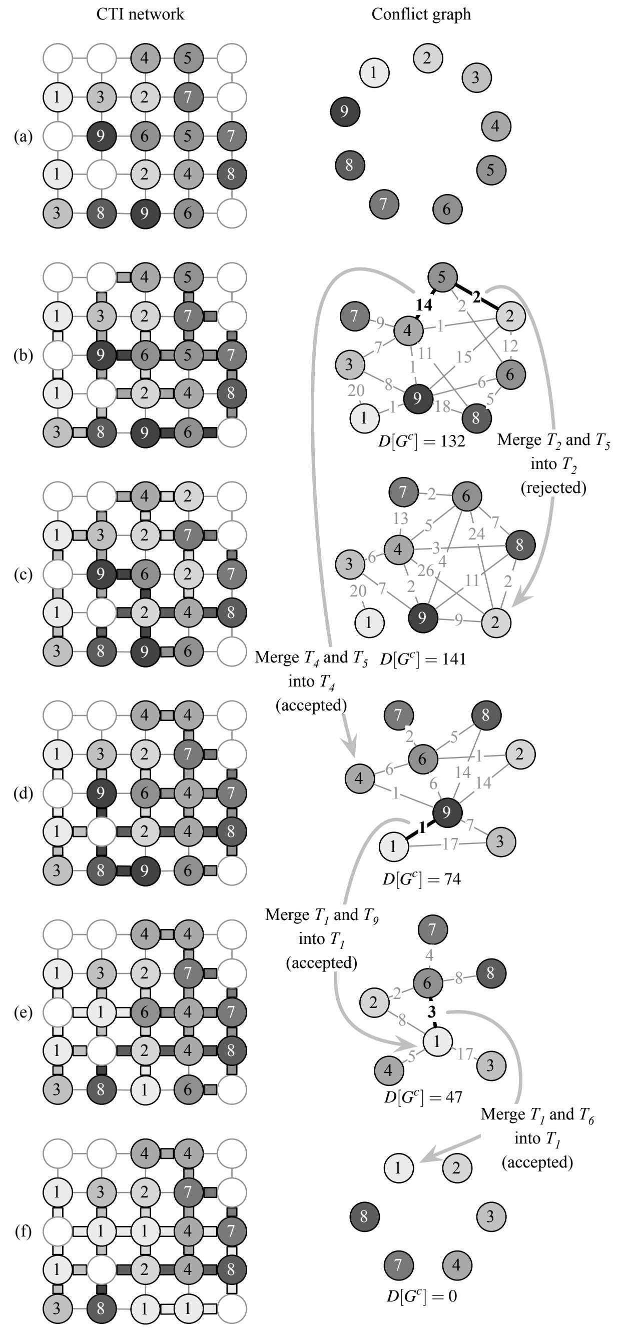

CTI network
Conflict graph
(a)
(b)
(c)
(d)
(e)
(f)
$D[G^c] = 132$
Merge T_2 and T_5 into T_2 (rejected)
$D[G^c] = 141$
Merge T_4 and T_5 into T_4 (accepted)
$D[G^c] = 74$
Merge T_1 and T_9 into T_1 (accepted)
$D[G^c] = 47$
Merge T_1 and T_6 into T_1 (accepted)
$D[G^c] = 0$

Figure A2: Detailed steps of the CTI routing of Figure 6.