Hardware and Software Requirements for Implementing a High-Performance Superconductivity Circuits-Based Accelerator

Mehdipour, Farhad
E-JUST Center, Kyushu University

Honda, Hiroaki
Department of Informatics, Kyushu University

Inoue, Koji
Department of Informatics, Kyushu University

Murakami, Kazuaki
Department of Informatics, Kyushu University

出版情報：The 3rd Asia Symposium on Quality Electronic Design, 2011-07. IEEE
バージョン：
権利関係：

Hardware and Software Requirements for Implementing a High-Performance Superconductivity Circuits-Based Accelerator

Farhad Mehdipour[1], Hiroaki Honda[2], Koji Inoue[2], Kazuaki Murakami[2]
[1]E-JUST Center, Kyushu University, Fukuoka, JAPAN
[2]Department of Informatics, Kyushu University, Fukuoka, JAPAN
[1]farhad@ejust.kyushu-u.ac.jp

Abstract

Single-Flux Quantum based large-scale data-path processor (SFQ-LSRDP) is a reconfigurable computing system which is implemented by means of superconductivity circuits. SFQ-LSRDP has a capability of accelerating data flow graphs (DFGs) extracted from scientific applications. Using an alternative technology instead of CMOS circuits for implementing such hardware entails considering particular constraints and conditions from the architecture and tools development perspectives. In this paper, we will introduce hardware specifications of the LSRDP and the tool chain developed for implementing applications. Placing and routing data flow graphs is a fundamental part to develop applications on the SFQ-LSRDP. Algorithms for placing DFG operations and routing nets corresponding to the edges of data flow graphs will be discussed in more details. These algorithms have been applied on a number of data flow graphs and the results demonstrate their efficiency. Further, simulation results demonstrates remarkable performance numbers in the range of hundreds of Gflops for the proposed architecture.

Keywords

Reconfigurable processors, single-flux quantum circuits, data flow graph, placement and routing

1. Introduction

Nowadays, reconfigurable processors provide a solution for high-performance computing systems, in which a reconfigurable accelerator can significantly relieve the burden of the main processor in computation-intensive part of applications. In [2], [6] and [13] examples of various high-performance solutions for specific application domains have been introduced. There are some serious issues in realizing powerful computing systems using recent finer CMOS technologies such as high heat radiation, difficulty in high-density packaging and etc. As an alternative to CMOS circuits, single-flux quantum (SFQ) circuits seem suitable due to featuring high-speed transmission, low-power consumption, small area as well as low-heat radiation. The basic component of SFQ digital circuits is a superconducting loop which carries information as a 1mV extremely low-width pulses in a very high speed (up to light speed), that can be implemented by Josephson junctions [7].

In [12] a single-flux quantum-based large-scale reconfigurable data-path processor (SFQ-LSRDP) has been introduced which is a reconfigurable processor consisting of a general purpose processor, a memory system and the LSRDP as an accelerator (Figure 1). LSRDP is consisting

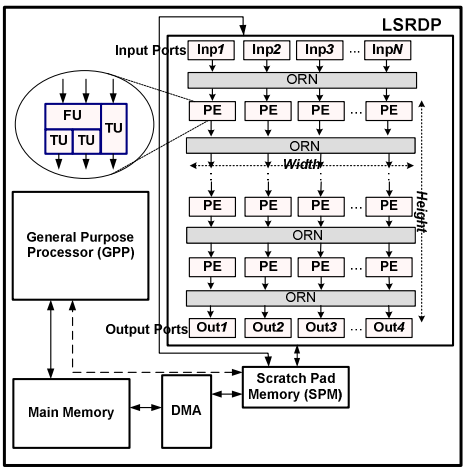

Figure 1. Overall architecture of the SFQ-LSRDP

of hundreds of processing elements (PEs) arranged in a pipelined fashion as well as operand routing networks (ORNs) which are providing interconnection resources between a pair of consecutive rows. The entire LSRDP architecture including PEs and ORNs are implemented by SFQ circuits instead of CMOS circuits.

Implementing target applications on the proposed hardware is performed through extracting data flow graphs (DFGs) of critical segments of target applications, mapping DFGs onto the PE array and generating corresponding reconfiguration bit-stream (for the LSRDP) as well as executable binary code (for the baseline processor). DFG nodes represent the operations and its edges appear as dependencies/connections among the operations/nodes. During execution of an application, configurations' bit-streams associated with the critical segments are loaded onto the LSRDP and executed to gain higher performance and likely lower energy consumption.

To the best of our knowledge, SFQ-LSRDP is the first one which is implemented using superconductivity circuits, and poses different constrains and conditions in the design and development phases, thus conventional solutions are not directly applicable. A design procedure has been introduced in [8] for the SFQ-LSRDP which considers the basic characteristics of the LSRDP architecture as well as constraints originated from the SFQ technology.

The main contribution of this paper is to highlight the hardware and software requirements for implementing SFQ-LSRDP and developing applications on it. In Section 2, a brief review on the LSRDP components and routing network architecture will be given. The tool chain will be

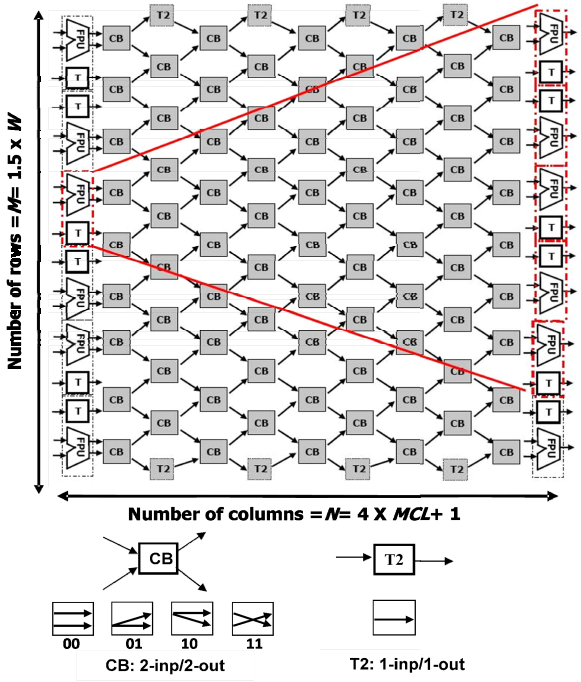

Figure 2. The ORN architecture

introduced in Section 3, including various components of the tool flow, interface between GPP and LSRDP and mapping tool as a main component of the tool chain. A number of placement and routing algorithms are discussed in Section 3 with more details as well. The experimental results are described in Section 4 and finally, Section 5 concludes the paper.

2. A brief view on the LSRDP architecture

PE and ORN arrays are two main components of the LSRDP architecture, while LSRDP dimensions represent the height (H) and width (W) of it as the number of rows and columns, respectively. Since, between each two consecutive rows, there is only one ORN, therefore, the total number of ORNs is H+1. A PE includes a functional unit (FU) for implementing desired operations (i.e. ADD/SUB and MUL) and a TU (transfer unit) as a routing resource. As ORNs provide routing resources only between consecutive rows, TUs are utilized to connect two PEs located on inconsecutive rows. Further, it is possible to use a FU for implementing two simultaneous transfer units (totally three TUs). A PE has three inputs and three outputs as well. I/O ports are located on top and bottom boundaries of the LSRDP as displayed in Figure 1.

During the design process of the LSRDP, various design alternatives such as different PE structures, different layouts representing the arrangement of various operations realizable by the PE array, granularity of functional units, total number of PEs, PE array dimension, ORN structure and etc. have been investigated. Efficiency of each instance of LSRDP architecture has been evaluated through a design space exploration approach [8]. Finally, an appropriate architecture meeting the design and implementation constraints and promising to achieve a higher performance has been developed. A number of scientific applications from [9] and [10] have been studied and attempted as benchmark applications within the design procedure. Analyzing the applications and their corresponding DFGs has been performed by means of the tools and algorithms which will be introduced in succeeding sections. These tools are modified and used in the application development process as well.

Also, an appropriate architecture should be developed for the ORN as the main routing resource for establishing connections between PEs. A typical ORN structure locating between two rows of PEs is displayed in Figure 2. The connection length of $PE_{i,j}$ (PE located in row i and col j) and $PE_{i+1,k}$ is the horizontal distance between them which can be calculated as $CL_{i,j,k}=|j-k|$. Correspondingly, the maximum connection length (MCL) is the maximum horizontal distance amongst all connections ($MCL= Max(CL_{i,j,k})$ $i=1..H-1$ and $j, k= 1..W$), while H and W are the LSRDP dimensions. MCL plays an important role as it affects the ORN size as well as the total LSRDP area.

The ORN architecture can be implemented by means of crossbar switches (CBs) [5] as shown in Fig. 2, where 2-inp/2-out CBs arranged in a chessboard pattern form an $M \times N$ ORN. The crossbar switches must be capable of multicasting of either of the inputs in addition to 'cross' and 'bar' functions. T2 is a crossbar switch with only one input, from which data can be directly sent to the output. The network structure consists of $1.5 \times W$ rows and $4 \times MCL + 1$ columns of CBs. The crossbar-based ORN has a regular pipelined structure that does not limit the performance of the LSRDP and can be reconfigured dynamically. Latency from input to output depends on the number of pipeline stages of the ORN which is equal to the number of CB columns. It can also be easily re-designed for any given complexity by adding a necessary number of extra rows of crossbars. Each PE in a LSRDP row can be connected to any of $2 \times MCL + 1$ PEs in the consecutive row via ORNs. Instead of multiple ORNs, only one ORN structure is implemented between every two rows. The ORN totally consists of $2 \times W \times 4 \times MCL$, and each CB needs around 550 Josephson Junctions (JJs) for implementation while each FU, i.e., ADD/SUB or MUL requires around 40 KJJs [5].

3. Tool chain

A dedicated tool chain has been developed for the SFQ-LSRDP, so that the input is a C code and the output is the configuration bit-stream file for the LSRDP as well as an executable code for the GPP. To interact GPP with the LSRDP a number of new instructions have been defined.

3.1. Overview

The main objective of the tool chain is to generate a configuration bit-stream file (path 1 on Figure 3) for the LSRDP and an executable code for the GPP (path 2 on Figure 3). The input to the tool chain is an application C code. This code is analyzed and the critical sections are extracted in the form of data flow graphs from the application code.

Generally, extracting data flow graphs from applications can be performed manually or automatically by means of a sophisticated high-level profiling tool. In the former case, programmer needs to have a sufficient knowledge on the

application and its detailed characteristics. On the other hand, automation of the hw/sw co-design methodology brings with it the need to develop sophisticated high-level profiling tools e.g. gprof [1], ProfileME [4], HALT [14].

Referring to path 1, firstly the input code is modified so that the special and predefined instructions for communicating to the LSRDP hardware are inserted in the code and the parts of code corresponding to the critical sections are eliminated. The modified code is compiled using COINS as a compiler infrastructure for the target GPP. Final executable code and the configuration files can be used as an input to a simulator which is employed for performance evaluations. A part of compiler tools can be customized for utilizing in the LSRDP design phase as well.

According to path 2, DFGs are mapped onto the LSRDP through locating DFG nodes on the PEs, routing interconnections as well as positioning input/output nodes on the proper I/O ports. The LSRDP specifications should be considered during the mapping process. Configurations' bit-stream corresponding to each one of DFGs can be generated after completion of the mapping stage.

3.2. Interface to the LSRDP

Dedicated instructions for managing transactions between GPP and the accelerator ought to be defined and used in the code development targeting the SFQ-LSRDP. GPP should interact with LSRDP during executions by managing control signals to transfer required input/output data from/to memory to/from the stream buffers, transmit configuration data from memory to the LSRDP for reconfiguration, invocate the LSRDP to start operating, synchronize between LSRDP and GPP and so on. Table 1 describes the list of instructions and functionality of each one briefly.

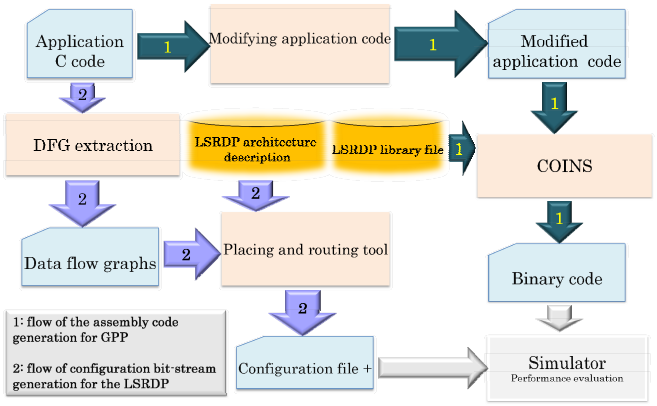

Figure 3. The tool chain

Table 1. LSRDP instruction set

Instruction	Description
conf_lsrdp	configures the LSRDP's data path
set_io_info_lsrdp	sets data to be read as inputs of the LSRDP in several consecutive iterations
run_lsrdp	runs the LSRDP (***non blocking*** execution)
Sync_lsrdp	GPP waits for the RDP to finish its task
set_IO_data_seq_lsrdp	sets memory pointer to the area that LSRDP should read from

3.3. Mapping tool

Mapping data flow graphs onto the accelerator includes two sub-problems i.e. placement and routing. Throughout the mapping process, DFG nodes are placed on appropriate positions (PEs) on the LSRDP. This is similar to the well-known placement problem, however the main goal in developing applications on the LSRDP is to minimize the *MCL* size and the LSRDP total area as well.

Routing process establishes connections between source and target PEs in the LSRDP by means of routing resources including ORNs and transfer units. It is supposed that each PE can implement one or a couple of transfer units for passing data to inconsequent rows. For each connection it is aimed to find a path between the source and destination with minimized *MCL*.

3.3.1. Placement

Placing DFG nodes is performed in three steps including the input nodes placement, operations placement and output nodes placement. Input/output nodes of the DFG should be located on appropriate input/output ports of the LSRDP on top/down boundaries. ORNs as routing resources exist between the first/last LSRDP rows and input/output ports. The main objective is to reduce the horizontal connection legnth between input/output ports and PEs in the first/last row of the LSRDP. Since DFG nodes are placed based on the location of their parents inside the LSRDP, placing input nodes is perfomed in a different manner from the placing output ports and it has more imapct in the quality of final placement. Propoer locations for output nodes can be determined based on the position of parent nodes which have already been placed.

In [8] a simple placement algirthm for input nodes referred as *fan-out based placement*, I/O nodes are placed with respect to their fan-out or the total number of children. First, input nodes of DFG are prioritized with repsect to their fan-out numbers, and then the placement algorithm looks for propoer locations for them over the input ports to minimize the longest connection length. We have also proposed a different algorithm referred as *proximity factor-based placement* for input nodes. The main intuition behind the proposed algorithm is to locate input nodes which have stronger connections to each other in a closer distance. Proximity factor is representing the strength of forces that input ports can exert to each other depending upon the connectivity of their descendants in sub-trees. This algorithm is extended to locate DFG nodes on the PEs as well.

For locating DFG nodes on the PEs we propose a naive alogrithm which tries to find a suitable position for each node with respect to the positions of its parents. Obviously, due to avaiablity of only unidirectional routing resources from each row to its conseuent row, each node should be placed in the lower rows where the parents are in upper rows of array. This process is run after the input nodes plancement, therefore starting from the nodes with the lowest ASAP (As Soon As Possible) level, their position are determined, afterward this process is carried on for the next levels. To place each node, first an initial row is determined,

which is the row next to the lowest row wherein a parent of intended node is located. Starting with the initial row, every unoccupied PE is examined in terms of the connection length criterion. For each PE which is being attempted, the following term is caluculated:

$$TCL = \sum_{i=1}^{n} \left| \frac{d_h^i}{d_v^i} \right|$$

where, $\frac{d_h^i}{d_v^i}$ is the horizontal/vertical distance of PE assigned to *i*-th parent of the node to the PE which is being examined, and *n* is the number of parents. After calculating *TCL* for every PE in the initial row and its succeeding rows, the PE with minimum amount of *TCL* is chosen as the place for locating the DFG node. If there would be more than one PE with the mimimum *TCL*, among them a PE with mimimum total horizontal distances to its parents will be chosen.

3.3.2. Routing

Routing process searches a route between source and target PEs in the LSRDP in two steps. First, each net connecting a pair of PEs is globally routed using available resources including ORNs and TUs. Afterward, a micro-routing algorithm finds paths through the cross-bar switches for the nets connecting ORN's inputs to outputs.

Routing global nets:

The input of routing algorithm is a netlist representing a list of interconnections (nets) between DFG nodes as well as placement information including the position of nodes in the LSRDP array.

According to the underlying LSRDP architecture, graph model for routing can be presented as Figure 4 in which the vertices and edges are representing the PEs and interconnection resources, respectively. The edges are two-terminal and unidirectional. We suppose that $G=(V,E)$ is the graph to describe the LSRDP routing layout. (s, d) is a global net which should be routed between source (v_s) and destination (v_d) vertices in the graph. A path $P_{s,d}$ is constructed through the routing algorithm, and consists of a sequence of vertices represented as $P_{s,d} = (v_s = v_{s,d}^0, v_{s,d}^1, v_{s,d}^2, ..., v_{s,d}^{D_v(v_s,v_d)-1}, v_d = v_{s,d}^{D_v(v_s,v_d)})$, while $D_v(v_s,v_d)$ is the vertical distance between source and destination vertices and $v_{s,d}^i$ ($i \in \{0, ..., D_v(v_s,v_d)-1\}$) is an intermediate vertex corresponding to the *i*-th hop of the path. Obviously, vertical distance of two consecutive vertices on the path (e.g. $v_{s,d}^i, v_{s,d}^{i+1}$) is equal to one. $D_h(v_1,v_2)$ is defined as the horizontal distance of two vertices v_1 and v_2. A capacity is defined for every edge and vertex as well. It is initiated to 1 for all edges, however the capacity of vertices varies from 1 to 3 based on the number of available transfer units at the corresponding PE (each vertex in the graph is associated with a PE in the LSRDP array within placement process). As aforementioned, every PE can implement from 1 (when FU is used for implementing an operation) up to 3 TUs (in case of availability of both FU and TU). The initial

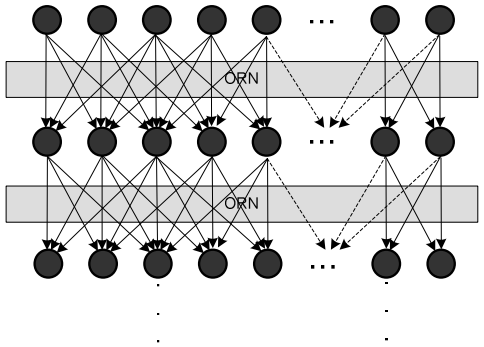

Figure 4. Routing graph model for the LSRDP when *MCL*=2

capacities of vertices are assigned after the placement, and alter during the routing process. Once a net is routed, the capacities of involved vertices and edges decrease by 1.

For each net (s, d), a target connection length is defined as $D_{Ts,d}=D_h(v_s,v_d)/D_v(v_s,v_d)$. The objective of the proposed routing algorithm is to minimize the deviation of horizontal connection lengths from $D_{Ts,d}$:

$$Minimize \sum_{i=0}^{D_v(v_s,v_d)} \left| D_{Ts,d} - D_h(v_{s,d}^i, v_{s,d}^{i+1}) \right|$$

Two algorithms has been proposed for global routing. The first routing algorithm is an iterative procedure of finding path between source and destination PEs. All DFG edges are dealt with as two-terminal nets even if some of them have common source or destinations. A modified maze router runs so that several paths start at the source, and are expanded until one of them reaches the target. Afterward, all the employed resources on the path are labeled as used ones which are no longer available for the next routes. Therefore, a higher priority is given to the critical nets which are located on DFG's critical paths.

A main difference with the traditional maze routers [11] is that ours tries to make the horizontal connection length closer to D_T at each step rather than minimizing the connection lengths. The time complexity of this algorithm is $O(MCL^H)$, which is indicating an exponential growth, hence a significant execution time for the long nets.

The second routing algorithm which is referred as quick router addresses the long execution time that the first algorithm incurs. Starting from the source, for each PE satisfying the *MCL* size constraint, summation of horizontal distances to source and destination PEs is calculated and a PE with minimum value is chosen as intermediate node an it will be considered as a source PE for the next iteration as well. This is repeated for the consequent rows until reaching to the destination. When a route fails due to lack of transfer node or violating *MCL* constraint, the algorithm backtracks to an upper row by choosing another PE which has the smallest connection lengths to source and destination. This algorithm falls in the category of branch and bound algorithms while the former one tries all possible routes exhaustively. The latter algorithm demonstrates a very short routing time in the range of seconds for the longest routes. Figure 5 displays how two algorithms work for a given net connecting a pair of source and destination PEs.

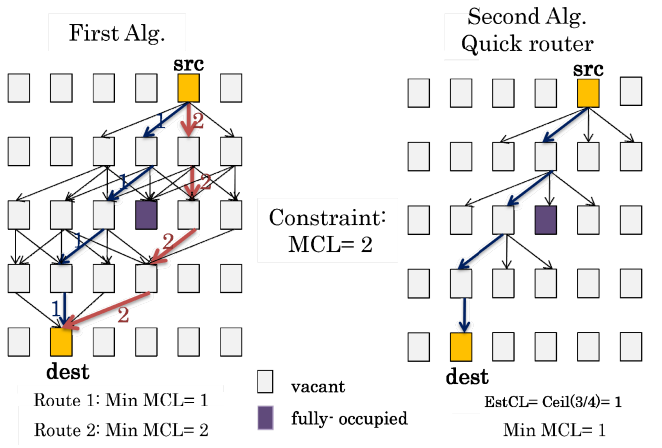

Figure 5. Routing graph model for the LSRDP when *MCL*=2

In the first algorithm all possible paths are searched exhaustively to find the best route meeting the constraints as well as satisfying the abovementioned goal. Two of possible routes have been indicated in the figure. Second algorithm only tries the best possible connection among several ones in each step and continues till reaching to destination. Other routes are tried when one fails.

Micro-routing algorithm for the ORN:

The aim of micro-routing algorithm is to route the nets through the CBs inside ORN. For each net passing through the ORN (referred as micro-net), CBs are configured to create a path from one input to one or more outputs of the ORN.

Here are some definitions:

- ORN^k: *k*-th ORN located between LSRDP rows *i* and *i+1*.
- $\{I_1^k, I_2^k, ..., I_{2\times M}^k$: (indexes of) the inputs of ORN^k.
- $\{O_1^k, O_2^k, ..., O_{2\times M}^k$: (indexes of) the outputs of ORN^k.
- $CB_{i,j}^k(inp_1)$, $CB_{i,j}^k(inp_2)$: inputs of the CB located in row *i* and column *j* of the ORN^k.
- $CB_{i,j}^k(out_1)$, $CB_{i,j}^k(out_1)$: outputs of the CB located in row *i* and column *j* of the ORN^k.

The inputs to this algorithm are as follows:

- LSRDP array dimensions (*W* and *H*).
- ORN dimensions including the number of CB rows and the number of CBs in each row (i.e. *M*, *N*).
- For each ORN^k, *k= 1..H-1*, list of micro-nets: $\{(I_i^k, O_j^k) | i, j \in \{1,2,...,2\times M\}\}$, the micro-net connecting I_i^k to O_j^k }.

The output of routing algorithm is the list of paths so that each path includes a list of CBs as well as configuration of each CB located on the path.

We propose a naïve algorithm for solving the above routing problem. Firstly, for each ORN, the outputs of CBs

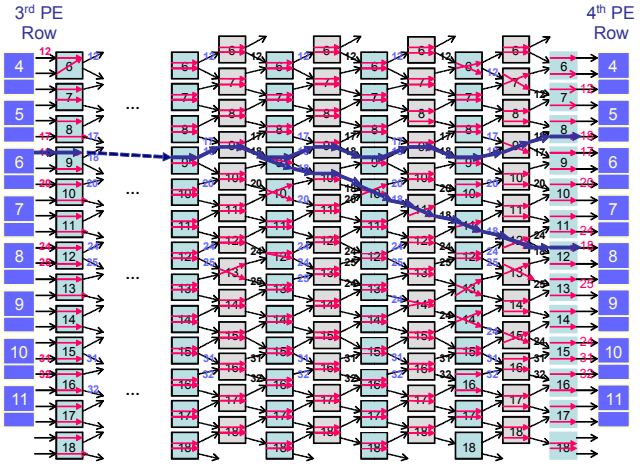

Figure 6. An example of ORN micro-routing for Heat 8x2

located in the last column are being labeled with the index of the inputs of corresponding nets. Unlabeled CB outputs are labeled with -1. Through a back-tracking algorithm, inputs of CBs in each column are labeled with respect to the CBs' output labels until reaching to the first column. A path will be easily recognized by grouping CBs with similar labels. The time complexity of this algorithm is $O(M \times N)$. Figure 6 shows a result of routing on a piece of the ORN located between 3rd and 4th rows of the LSRDP for a DFG extracted from Heat-8x2 [10] application. The micro-nets should be routed are: $\{(I^3_{12}, O^3_{14}), (I^3_{17}, O^3_{18}), (I^3_{18}, O^3_{17}), (I^3_{18}, O^3_{24}), (I^3_{20}, O^3_{20}), (I^3_{24}, O^3_{23}), (I^3_{24}, O^3_{30}), (I^3_{25}, O^3_{26}), (I^3_{31}, O^3_{31}), (I^3_{32}, O^3_{32})\}$. For example, according to this netlist, input 18 of ORN should be connected to outputs 17 and 24. The path starting from input 18 splits at the middle; therefore two separate paths reach to the outputs 17 and 24 from the branch in the middle.

It can be proved that using the proposed ORN architecture, every output of the ORN is reachable from any input located in the horizontal distance up to *MCL*. It should be noted that the nets' congestion for ORNs varies from upper rows of the LSRDP, where ORNs are highly congested comparing with the lower rows, where they are usually sparse.

ORN Micro-routing algorithm:
for each *k= 1* to *H-1* (routing micro-nets for each ORN)
  for every net (I_j^k, O_j^k) of ORN^k

$CB^k_{\lceil O_j^k/2 \rceil_N}(Out_{1-O_{j\%2}^k}) = I_j^k$ //output of the CB with the output O_j^k is being labeled with I_i^k.

  for each column *c* of ORN^k, *c = N* to *1*
    for *r*-th CB in column *c* (i.e. $CB_{i,j}^k$), *r= 1 to M*
      if $CB_{r,c}^k(out_1) < CB_{r,c}^k(out_2)$

$CB_{r,c}^k(inp_1) = CB_{r,c}^k(out_1), CB_{r,c}^k(inp_2) = CB_{r,c}^k(out_2)$

      else

$CB_{r,c}^k(inp_1) = CB_{r,c}^k(out_2), CB_{r,c}^k(inp_2) = CB_{r,c}^k(out_1)$

4. Experiment results

To demonstrate the efficiency of the proposed architecture and algorithms a number of experiments were

conducted. Four various calculations were attempted as scientific benchmark applications including: one-dimensional heat (referred as Heat) and vibration equations (Vibration), two-dimensional Poisson equation (Poisson) [10], and recursion calculation part of electron repulsion integral (ERI [9]) as a quantum chemistry application. Calculations consist of only ADD, SUB, and MUL operations and the DFGs are extracted manually from the applications. Table 2For each DFG the total number of nodes, the number of operations as well as the number of input/output nodes have been displayed in Table 2. For applications Heat and Vibration, more than one DFG as expanded versions of the basic DFGs have been generated. Also, two DFGs from ERI are attempted in our experiments.

The experiments are accomplished on a machine with Intel core i7 CPU 870@2.93GHz and 8GB RAM. Table 3 shows the routing results. Average horizontal and vertical connection lengths, maximum vertical connection lengths as well as the number of global and micro nets have been shown in the Table. The last column denotes the mapping time including the time spent for the placement and routing global and micro-nets. Total mapping time highly depends on the number of global nets, the average vertical length, and in particular the maximum vertical length of global nets. The time spent for placing and micro-routing is trivial for all DFGs. On the other hand, the number of vertices explored during the routing process grows exponentially for the long global nets, hence leads to a too time-consuming process. For example, in Poisson-3x3, existing only one long net with the length equal to 16 results in substantial increase in routing time. Figure 7 shows a schematic view of the mapping result for Vibration-8x2. The connections in red are indicating the *MCL* that is equal to 2 for this DFG.

Table 2. Specifications of the extracted DFGs

DFG	total # of nodes	# of inputs	# of outputs	# of ops
Heat-8x1	34	6	4	16
Heat-8x2	60	8	4	32
Heat-16x2	172	16	12	96
Poisson-3x3	62	18	1	33
Vibration-4x2	48	8	4	24
Vibration-8x2	136	16	12	72
Vibration-8x4	168	16	8	96
ERI-1	76	16	9	51
ERI-2	67	19	1	47

Table 3. Results of routing nets using the proposed algorithms

DFG	avg. hor. C.L.	avg./max. ver. C.L.	# of global/micro nets to route	Time to map (sec)
Heat-8x1	0.35	0.75/3	36/64	0.015
Heat-8x2	0.44	1.32/5	68/114	1.75
Heat-16x2	0.47	1.64/7	204/343	1.05
Poisson-3x3	0.68	2.4/16	67/120	2074.5
Vibration-4x2	0.46	1.58/9	50/88	0.34
Vibration-8x2	0.42	2.15/10	154/332	2.20
Vibration-8x4	2.48	3.72/16	348/610	6721.3
ERI-1	0.75	2.21/9	111/374	53.61
ERI-2	0.78	2.99/9	95/332	0.327

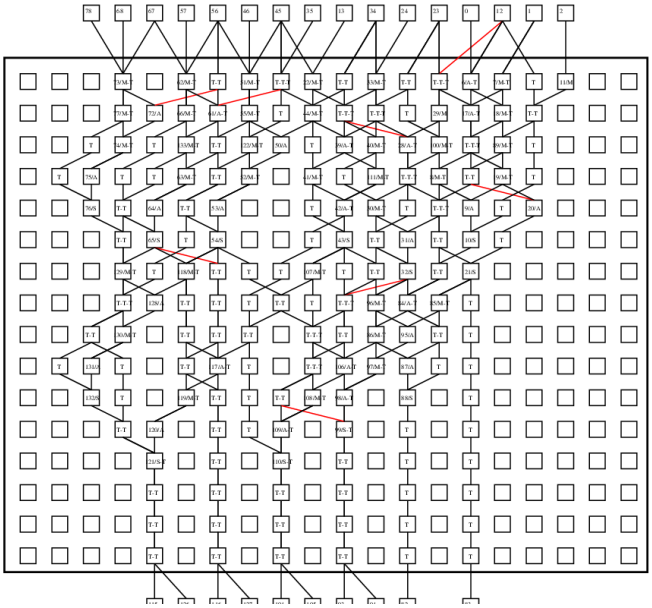

Figure 7. Result of mapping for Vibration-8x2

Using the quick router proposed in Section 3 which is relying on a branch and bound algorithm, the above issue is resolved and the routing time is remarkably reduced to a few seconds in the worst case. According to the experiment results, by using quick router the routing time is around 3 sec for Vibration-8x4 which has several number of long connections. For other benchmarks, total mapping time is under 1 sec. Further, quick router maintains efficiency of the basic algorithm as the *MCL* size remains unchanged.

Currently, fabricating only small scale LSRDP architecture is possible however, implementing large-scale accelerator using SFQ circuits is the main target. We have obtained preliminary performance evaluation results through a simulation for two applications, i.e., Heat and Vibration. Experiments show by using the LSRDP machine including a 3.2 GHz out-of-order base processor and operating frequency of 80 GHz for the LSRDP, remarkable performance values, namely 210 Gflops and 104.9 Gflops are achievable, respectively. Moreover, discovering new applications promising for gaining high performance numbers in the range of Tera-flops is another challenge which will be addressed in our future work.

5. Conclusion

We discussed about developing a high-performance low-power computer which is composed of a GPP and an accelerator. The proposed hardware accelerator is a large-scale reconfigurable data-path computing unit with a large matrix of PEs implemented by SFQ circuits, which makes it suitable for executing massive computational-intensive scientific applications. LSRDP architecture has been designed through a quantitative analysis on the quality of target applications. A mapping procedure is a major part of the developed tool chain that can be exploited both in design and utilization stages. According to the observations from simulation results, LSRDP is promising to achieve

noticeable performance values in the range of hundreds of Gflops.

Acknowledgment

This research was supported in part by Core Research for Evolutional Science and Technology (CREST) of Japan Science and Technology Corporation (JST).

References

[1] B. Cmelik, SpixTools introduction and user's manual, Techincal Report SMLI TR-93-6, Sun Microsystems Laboratory, Mountain View, CA, February 1993.

[2] E. Cho, and G. Bourgeois, Efficient and accurate FPGA-based simulator for molecular dynamics, IEEE Int'l Symp. on Parallel and Distributed Processing (IPDPS), pp. 1-7, 2008.

[3] COINS Compiler Infrastructure, http://www.coins-project.org/international.

[4] J. Dean, J. Hicks, C. Waldspurger, W. Weihl and G. Chrysos, ProfileMe: Hardware support for instruction-level profiling on out-of order processors, In Proceedings of International Symposium on Microarchitecture, 1997.

[5] Kataeva et al., An operand routing network for an SFQ reconfigurable data-path processor, *IEEE Trans. Appl. Supercond.,* vol. 19, no. 3, pp. 665-669, 2009.

[6] Y. Komeiji, et al., Fast and accurate molecular dynamics simulation of a protein using a special-purpose computer, *Journal of Computational Chemistry*, 18(12): pp. 1546-1563, 1997.

[7] K. Likharev and V. Semenov. RSFQ logic/memory family: a new Josephson junction technology for sub-terahertz clock frequency digital systems. *IEEE Trans. on Appl. Supercond.*, vol. 1, no. 1, pp. 3-28, 1991.

[8] F. Mehdipour, H. Honda, K. Inoue, H. Kataoka, and K. Murakami, A design scheme for a reconfigurable accelerator implemented by single-flux quantum circuits, *J. Syst. Architect.* (2010), doi:10.1016/j.sysarc.2010.07.009.

[9] S. Obara and A. Saika, Efficient recursive computation of molecular integrals over Cartesian Gaussian functions, *J. Chem. Phys.,* vol.84, pp.3963, 1986.

[10] W.H. Press, B.P. Flannery, S.A. Teukolsky, and T.W. Vetterling, Numerical recipes in C, Cambridge University Press, 1988.

[11] N. Sherwani, Algorithms for VLSI physical design automation, Third Edition, Kluwer Academic Publishers, 1999.

[12] N. Takagi, K. Murakami, A. Fujimaki, N. Yoshikawa, K. Inoue, H. Honda, Proposal of a desk-side supercomputer with reconfigurable data-paths using rapid single flux quantum circuits, *IEICE Trans. on Elec.*, E91-C(3):350-355, 2008.

[13] S. Toyoda, et al., Development of MD engine: high-speed accelerator with parallel processor design for molecular dynamics simulations, *Journal of Computational Chemistry*, 20(2): pp. 185-199, 1999.

[14] C. Young, The Harvard atom like tool manual (HALT), http://citeseer.nj.nec.com/121315.html.