**ROUTING ARCHITECTURE AND ALGORITHMS FOR A SUPERCONDUCTIVITY CIRCUITS-BASED COMPUTING HARDWARE**

Farhad Mehdipour¹, Hiroaki Honda², Hiroshi Kataoka², Koji Inoue², Kazuaki Murakami²

¹E-JUST Center, Kyushu University, Fukuoka, JAPAN
²Department of Informatics, Kyushu University, Fukuoka, JAPAN

**ABSTRACT**

Dedicated tools for placing and routing data flow graphs extracted from computation-intensive applications are basic requirements for developing applications on a large-scale reconfigurable data-path processor (LSRDP) implemented by superconductivity circuits. 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. The main contribution of this work is to introduce an operand routing network (ORN) architecture as well as algorithms for routing the nets corresponding to the edges of the data flow graphs. Further, a micro-routing algorithm is proposed for routing and configuring the ORNs internally. These algorithms have been applied on a number of data flow graphs from target applications and the results demonstrate their efficacy.

Index Terms— Reconfigurable accelerator, single-flux quantum circuits, data flow graph, placement and routing.

**1. INTRODUCTION**

Complicated computations and simulations necessitate providing high computational power for individual researchers. Hybrid reconfigurable processors provide one 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 [1],[3] and [10] examples of various high-performance solutions for specific application domains have been introduced. There are some serious barriers in realizing powerful computing systems using recent finer CMOS technologies. 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 with Josephson junctions [4].

A reconfigurable processor consisting of a general purpose processor, a memory system and a single-flux quantum-based large-scale reconfigurable data-path processor (SFO-LSRDP) as an accelerator has been introduced in [9] (Fig 1). LSRDP has a pipelined architecture comprising a two-dimensional array of processing elements (PEs) and operand routing networks (ORNs) such that one PE can be connected through ORNs to a number of PEs in the next row. The entire LSRDP architecture including PEs and ORNs are implemented by SFQ circuits as a substitute to CMOS circuits. I/O ports are located on top and bottom boundaries of the LSRDP as displayed in Fig 1. LSRDP dimensions represent the height (H) and width (W) of it as the number of rows and columns, respectively. The PE architecture 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.

To boost the performance of an application, firstly data flow graphs (DFGs) are pulled out from critical segments of the application and configuration bit-streams corresponding to the DFGs which are generated by using a dedicated tool. DFG nodes represent the operations and its edges appear as dependencies/connections among the operations/nodes. During execution of an application, configurations associated with the critical segments are loaded onto the LSRDP and executed for achieving higher performance and lower power consumption.

To the best of our knowledge the SFQ-LSRDP is the first one which is implemented using superconductivity circuits instead of CMOS circuits, and poses different constrains and conditions in the design and development phases, thus entails solutions different from the traditional ones in some aspects. A design procedure has been introduced in [5] for the SFQ-LSRDP which considers the basic characteristics of the LSRDP architecture as well as constraints originated from the SFQ technology. Implementing target applications on the proposed hardware is performed through extracting data flow graphs, mapping them onto the PE array and generating corresponding reconfiguration bit-stream (for the LSRDP) and executable binary code (for the baseline processor).

![Fig 1. Overall architecture of the SFQ-LSRDP](image-url)
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.

Routing process finds the routes between source and target PEs in the LSRDP in two steps. First, each net connecting a pair of PEs is globally routed using the 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.

3.1. 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 Fig. 3 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_i) 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 

\[ P_{s,d} = (v_0 = v_s, v_1, v_2, ..., v_{i-1}, v_i, v_{i+1}, ..., v_d = v_d) \]

While 

\[ D_i(v_s,v_d) \]

is the vertical distance between source and destination vertices and 

\[ D_{\text{hsd}}(v_{i-1},v_i) \]

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. \( v_{i-1}^{\text{hsd}}, v_i^{\text{hsd}} \)) is equal to one. 

\[ D_i(v_s,v_d) \]

is defined as the horizontal distance of two vertices \( v_s \) and \( v_d \). A capacity is defined for each 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 capacities of vertices are assigned after the placement, and after during the routing process. Once a net is routed, the capacity of involved vertices and edges decreases by 1.
For each net \((s, d)\), a target connection length is defined as \(D_{s,c}=D_h(v_s,v_d)/D_h(v_s,v_d)\). The objective of the proposed routing algorithm is to make the entire horizontal connection lengths closer to \(D_{s,c}\):

\[
\text{Minimize } \sum_{i=0}^{n} \left| D_{s,c} - D_h(v_s,v_d) \right|
\]

The proposed 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 [8] is that ours tries to make the horizontal connection length closer to \(D_{s,c}\) at each step rather than minimizing the connection lengths. In Fig 4, it is shown that minimizing the connection length greedily in the first hops might result in violating MCL constraint in the latest hops. In this figure it is assumed that \(MCL=2\). For the first route denoted by \(r1\) on the edges, a greedy procedure is followed to minimize the connection length in each step, but in the last step while the connection length is equal to 3, the constraint on the MCL size is not met. Therefore, this route is not valid. Other two routes displayed in the figure as \(r2\) and \(r3\) are valid routes since they satisfy the constraint. The time complexity of this algorithm is \(O(M\times N)\), which is indicating an exponential growth, hence a significant execution time for the long nets.

### 3.2. 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 \(i+1\).
- \(I_1^k, I_2^k, ..., I_{M^k}\) : (indexes of) the inputs of \(ORN^k\).
- \(O_1^k, O_2^k, ..., O_{2M^k}\) : (indexes of) the outputs of \(ORN^k\).

- \(CB^k_{i,j}(inp_1), CB^k_{i,j}(inp_2)\) : inputs of the CB located in row \(i\) and column \(j\) of the \(ORN^k\).
- \(CB^k_{i,j}(out_1), CB^k_{i,j}(out_1)\) : 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 \((W, H)\).
- 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:

\[
\{(I^k_j, O^k_j), j \in [1,2,2 \times M]\} \text{' the micro-net connecting } I^k_j \text{ to } O^k_j\}
\]

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 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 \(O(M\times N)\). Fig. 5 shows a result of routing on a piece of the ORN located between \(3^{rd}\) and \(4^{th}\) rows of the LSRDP for a DFG extracted from Heat-8x2 [15] application. The micro-nets should be routed: \(\{(I^1_j, O^1_j), (I^2_j, O^2_j), (I^3_j, O^3_j), (I^4_j, O^4_j), (I^5_j, O^5_j), (I^6_j, O^6_j), (I^7_j, O^7_j), (I^8_j, O^8_j)\}\). For example, according to the 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 to the lower rows, where they are usually sparse.

**ORN Micro-routing algorithm:**

for each \(k = 1 \text{ to } H-1\) (routing micro-nets for each ORN)

for every net \(\{I^k_j, O^k_j\}\) of \(ORN^k\)

\[
CB^k_{\left| O^k_j/2 \right|, N}(Out_{\left| O^k_j/2 \right|}) = I^k_j \quad \text{/output of the CB with the output } O^k_j \text{ is being labeled with } I^k_j.
\]

for each net \(\{I^k_j, O^k_j\}\), following steps on \(ORN^k\) are repeated

for each column \(c\) of \(ORN^k\), \(c = N \text{ to } 1\)

// back-tracking from the last column

for \(r\-th\ CB in column \(c\) (i.e., \(CB^k_{r,c}(out_{1})\)), \(r = 1 \text{ to } M\)

if \(CB^k_{r,c}(out_{1}) < CB^k_{r,c+1}(out_{2})\)

\[
CB^k_{r,c}(inp_{1}) = CB^k_{r,c}(out_{1}), CB^k_{r,c}(inp_{2}) = CB^k_{r,c+1}(out_{2})
\]

else

\[
CB^k_{r,c}(inp_{1}) = CB^k_{r,c+1}(out_{2}), CB^k_{r,c}(inp_{2}) = CB^k_{r,c}(out_{1})
\]
4. EXPERIMENT RESULTS

To demonstrate the efficiency of the proposed architecture and algorithms a number of experiments were conducted, however due to the space limitation we only focus on a part of results. Four various calculations are attempted as scientific benchmark applications including: one-dimensional heat (referred as Heat) and vibration equations (Vibration), two-dimensional Poisson equation (Poisson), and recursion calculation part of electron repulsion integral (ERI [6]) as a quantum chemistry application. Calculations consist of only ADD, SUB, and MUL operations and the DFGs are extracted manually from the applications. The experiments are accomplished on a machine with Intel core i7 CPU 870 @ 2.93GHz and 8GB RAM. Table 1 shows the DFGs generated from above calculations, their specifications as well as the routing results. For applications Heat and Vibration, more than one DFG as expanded versions of the basic DFGs have been generated (more details in [5]). Also, two DFGs from ERI are studied in our experiments. For each DFG the number of operations as well as the number of input/output nodes have been displayed. 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 time required to perform routing for both global and micro-nets. Total routing 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 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. This issue will be addressed in our future work.

5. CONCLUSION

Routing operand network architecture based on cross-bar switches and two algorithms for routing global and micro nets were discussed in this paper. The algorithms rely on the traditional counterparts; however they have been customized for the SFQ-LSRDP to meet the requirements and constraints of the underlying hardware. Vertical distance from source to destination vertices of a global net is an important factor which strongly impacts the execution time of routing algorithm. In future, we will improve the algorithm so that long connections can be routed in a reasonably shorter time.

REFERENCES


