Developing an Architecture for a Single-Flux Quantum Based Reconfigurable Accelerator

Mehdipour, Farhad
School of Information Science and Electrical Engineering, Department of Informatics, Kyushu University

Honda, Hiroaki
Institute of Systems, Information Technologies and Nanotechnologies

Kataoka, Hiroshi
School of Information Science and Electrical Engineering, Department of Informatics, Kyushu University

Inoue, Koji
School of Information Science and Electrical Engineering, Department of Informatics, Kyushu University

他

出版情報：電子情報通信学会技術研究報告, ICD2009-111. 109 (405), pp.99-104, 2010-01. 電子情報通信学会
バージョン：
権利関係：

社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

Developing an Architecture for a Single-Flux Quantum Based Reconfigurable Accelerator

Farhad Mehdipour[1] Hiroaki Honda[2] Hiroshi Kataoka[1] Koji Inoue[1] and Kazuaki Murakami[1]

[1] School of Information Science and Electrical Engineering, Department of Informatics, Kyushu University, Fukuoka, Japan
[2] Institute of Systems, Information Technologies and Nanotechnologies, Fukuoka, Japan
Email: {farhad, dahon, kataoka}@c.csce.kyushu-u.ac.jp, {inoue, murkami}@ait.kyushu-u.ac.jp

Abstract As a solution to gain high performance computation, a large-scale reconfigurable data-path (LSRDP) processor is introduced in this paper. LSRDP is implemented by virtue of single-flux quantum circuits and integrated to a general purpose processor to accelerate the execution of data flow graphs (DFGs) extracted from scientific applications. Design procedure of the LSRDP and particularly the process of mapping DFGs onto the LSRDP are discussed and our techniques for optimizing the area of accelerator will be presented as well.

Keyword Reconfigurable accelerator, single-flux quantum, data flow graph, placement and routing

1. Introduction

Providing high computational power to individual researchers is crucial for progress of the research and development. Although, recent advances on chip design and fabrication provide the possibility for producing high-performance computers, there is still a high demand to meet the required performance for specific applications. As a solution, a hybrid architecture comprising a general purpose processor (GPP) and an accelerator can be exploited for special purpose computations. The accelerator should be designed such that can feature high performance, and low power consumption[1][2][6].

Undoubtedly, there are some serious barriers in realizing powerful computing systems using recent finer CMOS technologies. The most important issues are high heat radiation, long interconnection delays and memory-wall problem [10]. To overcome those barriers, a new architecture has been introduced which consists of a CMOS general purpose processor, a memory and a single-flux quantum (SFQ)- based Reconfigurable Large-Scale Data-Path processor (SFQ-LSRDP) as an accelerator (Fig. 1) [10]. The proposed architecture is expected to be a high-performance desk-side computer with low-power consumption and it is suitable for execution of scientific applications demanding massive computations.

A SFQ circuit relies on the superconducting technology which includes extremely lower power consumption and high-speed compared to the CMOS counterparts. A basic SFQ element uses a 1mV extremely low-width pulse as an information carrier that is propagated at very high speed (up to light speed) in the circuit. High-speed switching and signal transmission, low power consumption, compact implementation (small area), suitability for pipeline processing of data stream are the main features of the SFQ technology [5]. Since the LSRDP as a main component of the target architecture is implemented by SFQ circuits; obviously, it can address the abovementioned issues originating from CMOS technology.

Developing necessary tools for compiling applications, generating data flow graphs (DFGs) and configuration bit-streams as well as designing and fabricating the LSRDP architecture are the main phases of implementation of such computing system. In the architecture side, the main components of LSRDP as well as structure of routing resources which all are implemented by SFQ circuits will be discussed in this paper. In addition, we will concentrate on the DFG mapping tool as a basic component of the tool chain, and will describe how it is exploited during the design procedure of the LSRDP. Basic properties of the LSRDP architecture and constraints originating from the SFQ circuits ought to be taken into account within the tool development and LSRDP design procedure.

Placing input nodes on appropriate locations is a key step in the placement procedure which highly impacts the final cost in terms of on the total connection length. Connection length is also a key factor in turn which strongly influences the routing resources and overall area of the accelerator. We propose an algorithm based on proximity factor of input node pairs to reduce the maximum connection length. In addition, an alternative to connection length measurement is proposed that can be effective in reducing the implementation cost. We evaluate the proposed techniques during the LSRDP design procedure. Moreover, a benchmark of computational-intensive scientific applications are introduced which are attempted for designing the target hardware. The extracted DFGs demonstrate a huge flow of operations which needs a sophisticated mapping tool to map them onto the LSRDP and satisfy the constraints as well.

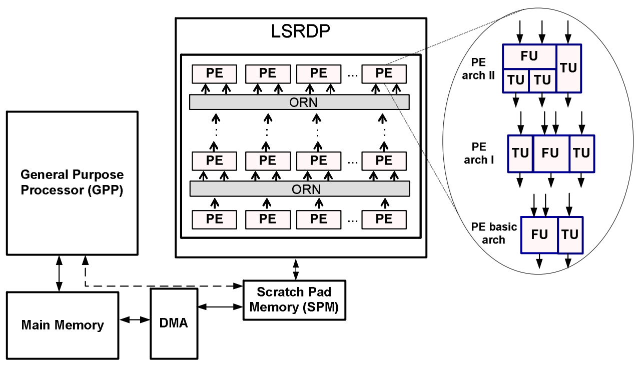


Fig. 1. Overall architecture of the SFQ-LSRDP computer

2. LSRDP General Architecture and Specifications

Generally, LSRDP is a pipelined architecture comprising a two-dimensional array of processing elements (PEs) such that one PE can be connected through operand routing networks (ORNs) to a number of PEs in the next row. Fig. 1 displays the overall architecture of the proposed high-performance computer consisting of a GPP, an LSRDP as accelerator and memory elements.

Data flow graphs are automatically or manually pulled-out from critical segments of applications and configuration bit-streams are generated by using a dedicated tool. During execution of an application, configurations associated with the critical segments are loaded onto the LSRDP and executed in favor of achieving higher performance and lower power consumption. Since the cascaded PEs can generate a final result without temporally memorizing intermediate data, the number of memory load/store operations corresponding to spill codes can be reduced. Therefore, memory bandwidth required to gain a high performance computation might decrease as well. Furthermore, as a loop-body mapped onto the PE array is executed in a pipeline fashion, LSRDP can provide a high computing throughput.

In the design procedure except the basic properties of the LSRDP architecture, it is intended to obtain the specifications of the architecture including following ones.

Layout: Layout of the LSRDP indicates the type of functional units and their distribution. Tow types of layout are examined for the LSRDP during the design procedure. In a normal layout (Layout-I), each FU can implement any operation including ADD/SUB and MUL. Layout-II is similar to a checker pattern, i.e. only one of operations ADD/SUB or MUL is implemented in each PE.

Input/Output ports: I/O ports are located on top and bottom boundaries of the LSRDP. The limitation on the number of ports depends on the available memory bandwidth, LSRDP operation frequency, width of data bus and the number of memory read/write channels.

LSRDP dimensions: Fig. 1 shows that LSRDP is a matrix of PEs in which the height and width of LSRDP are the number of rows and columns, respectively.

PE types: Three types of PE architecture are examined for the LSRDP (Fig. 1). Most suitable PE is selected during the design procedure. The basic PE architecture includes an FU for implementing desired operation and a TU (transfer unit). As ORNs provide only routing resources between consecutive rows, TUs are utilized to connect two PEs locating on inconsecutive rows. It is possible to use an FU for implementing a transfer unit as well. In addition, each PE has three inputs (two inputs for FU and one for the TU) and two outputs (one from FU and another from TU). The second PE architecture (PE arch. I) has one addioninal TU for increasing the flexibility of routing and it has 4-inps/3-outs. The third type of PE architecture (PE arch. II) has a similar architectue to the first one, the difference is in extending capability of implementing two simultaneous TUs by the FU (totally three TUs). An additional mux should be used inside the PE to choose between FU's output and the input.

Type and granularity of functional units: Each FU can implement basic 64-bit double-precision floating point operations e.g. ADD, SUB and MUL. Control instructions (branches) and direct memory accesses via PEs are not supported.

Operand routing network (ORN): PEs of each row are connected to the PEs in the next row through ORNs as routing resources. Fig. 2 shows the definition of the connection length and the maximum connection length (MCL) on a piece of LSRDP architecture. It can be seen that the connection length of two PEs is the horizontal distance of the PEs. Correspondingly, the MCL size is the maximum horizontal distance of two PEs located in two subsequent rows (Fig. 2). ORNs should provide all outputs of a PE with totally no_of_inputs x (2 x MCL+ 1) connections to the PEs in consequent row. ORNs' functionality is similar to a multiplexer however; ORNs are composed of cross-bar switches (CBs). Similar to other components of the LSRDP, CBs are also implemented by means of Josephson Junctions as the basic elements of the SFQ circuits [4].

An ORN for the PE architecture consisting of FUs and TUs is displayed in Fig. 3. The crossbar-based ORN has a regular pipelined structure that does not limit the performance of the LSRDP and can be reconfigured on the fly. It can also be easily re-designed for any given complexity by adding a necessary number of extra rows of crossbars [4].

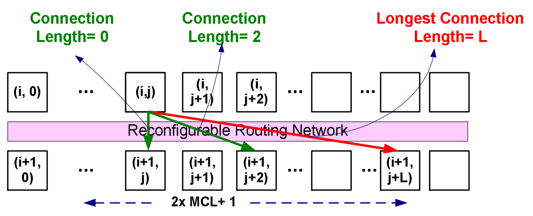


Fig. 2. Definition of the connection length and the maximum connection length (MCL)

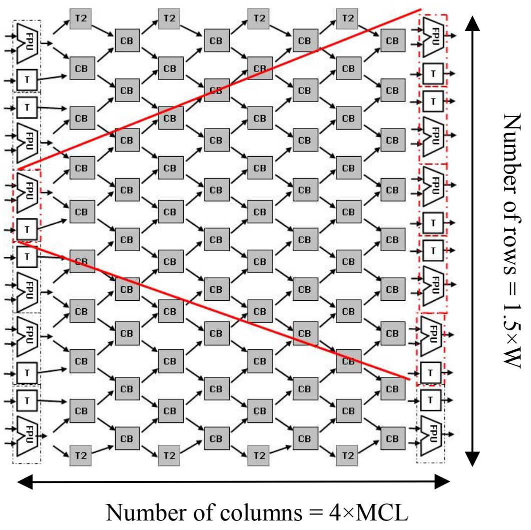


Fig. 3. Structure of a CB-based ORN for the PE basic arch. (MCL= 2)

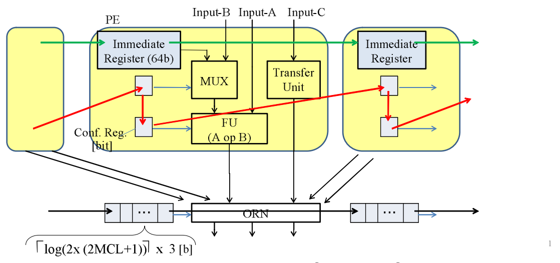


Fig. 4. Detailed architecture of a reconfigurable PE

Throughout the LSRDP architecture, a united ORN is implemented between each two rows rather than implementing separate ORNs for each PE. In Fig. 3 it is assumed that the MCL is equal to two, therefore, each PE in the left side (a row of PEs) can be connected to any of five (2x MCL+ 1) PEs in the right side (consecutive row). Two factors including the number of PEs in a row and MCL are the main factors which affect the ORN size. Moreover, the number of input/outputs for each PE is an important factor which can influence the ORN structure and its size. For the PE including one FU and one TU which totally has three inputs and two outputs, the ORN architecture is as Fig. 3. Assuming W as the number of PEs in each row, the ORN will totally consist of 2 x W x (4 x MCL) CBs.

Internal memory: 64-bit immediate registers are located in each PE in order to handle immediate values (Fig. 4).

Reconfiguration mechanism: LSRDP is a reconfigurable hardware that can be configured within run-time using the bit-stream generated for DFGs. Fig. 4 shows the architecture of a PE and how it can be reconfigured during the configuration phase. Apart initializing immediate registers, the multiplexers, PEs and ORN micro-routing network should also be programmed using the configuration bits. In order to configure each component, the configuration bit-stream is serially transferred to the configuration registers.

3. LSRDP Design Procedure and Mapping Tool

Entire design procedure is an iterative process of gathering statistics and analysis of results. We used a quantitative approach for designing the LSRDP architecture and determining its detailed architectural specifications. To determine each design parameter, DFGs should be mapped onto the LSRDP and the outcome is analyzed. The mapping process is performed without forcing any constraint except the constraints originated from the LSRDP architecture e.g. unidirectional data flow over the PE rows, availability of routing resources between subsequent rows and etc. In the next stage, the results of mapping should be analyzed by the designer to decide an appropriate value for the intended parameter.

3-1. DFG Extraction

Extracting data flow graphs from applications is 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. Four applications are attempted as scientific benchmark applications including: one-dimensional heat (referred as Heat) and vibration equations (Vibration), two-dimensional Poisson equation (Poisson) [8], and recursion calculation part of electron repulsion integral (ERI [7]) as a quantum chemistry application. All calculations consist of ADD, SUB, and MUL operations and DFGs have been extracted manually.

3-2. Placement and Routing

Mapping process consists of two sub-procedure i.e. placement and routing. Throughout the mapping process, DFG nodes are placed on appropriate positions (PEs) over the LSRDP. This is similar to the well-known placement problem [9]. Generally, minimizing the total connection length or the maximum connection length are main objectives, however in designing LSRDP, the mail goal is to minimize the maximum connection length (MCL) that directly impacts the ORN sizes and the LSRDP area as well.

Routing process is the next stage that establishes connections between the PEs in the LSRDP by means of routing resources including ORNs and transfer units [9]. As aforementioned, it is supposed that each PE can impelemt one or a couple of transfer units for routing data to inconsequent rows. For each connection it is aimed to find a shortest path between the source and destination PEs.

3-3. I/O Nodes Placement

Input/output nodes of the DFG should be assigned to appropriate input/output ports of the LSRDP on the boundaries (Fig. 5). Between the first/last LSRDP rows and input/output ports, ORNs are avaiable as routing resources. The main objective is to reduce the 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. Afterward, propoer locations for output nodes can be determined based on the position of parent nodes which have already been placed.

We propose two approaches for the input nodes placement.

1. fan-out based placement: I/O nodes are placed with respect to their fan-out or the total number of children. Input nodes of DFG are prioritized with repsect to their fan-out and then the placement algorithm looks for proper input ports to minimize the longest horizontal connection length. In this case, the DFG nodes should be initially placed onto the PEs prior to the input ports placement. Fig. 5 shows three cases (depending on the input node fan-out which might be one, two or more than two) for placing an input node on the ports. It is attempted to find the closest unoccupied port to the input node's children. DFG nodes should be remapped according to the location of input nodes.

2. proximity-factor based placement: placing input nodes considering solely the location of their direct descendants (in the closes vicinity in the DFG) might result in large MCLs as shown in Fig. 6. In a piece of mapping result displayed in Fig. 6, input nodes '0' and '24' are located in a close distance to their children (located in the first row) however they have a long distance to each other and to a common descendant node which is positioned in the third row as well. This placement without paying attention to the location of descendants make some long connections which affects the MCL size. One solution to cope with this issue is to locate input nodes which have strong connections to each other in a closer distance. We define a factor referred as proximity factor to represent the strength of forces that PEs can exert to each other. This factor is used as a basis of the input nodes placement algorithm.

Proximity factor: for each pair of inout nodes *i* and *j* the proximity factor is calculated as:

$$p_{i,j} = \sum_{k \in S_{i,j}} \frac{1}{D_{k,i}} \tag{1}$$

while $D_{k,i}$ is the distance of common descendant node *k* to the input ports *i* (and *j*). The distance of a node to its related input port can be calculated as its ASAP (As soon as posssible [3])

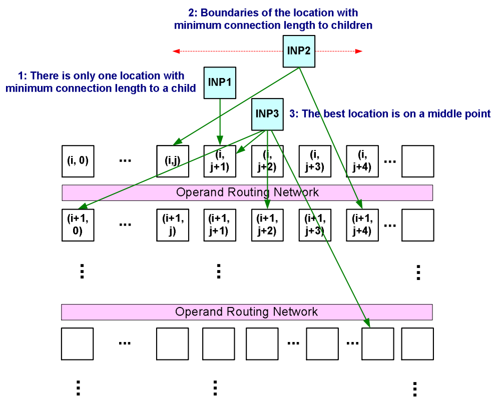

2: Boundaries of the location with minimum connection length to children
INP2
1: There is only one location with minimum connection length to a child
INP1
INP3
3: The best location is on a middle point
(i, 0)
(i,j)
(i, j+1)
(i, j+2)
(i, j+3)
(i, j+4)
Operand Routing Network
(i+1, 0)
(i+1, j)
(i+1, j+1)
(i+1, j+2)
(i+1, j+3)
(i+1, j+4)
Operand Routing Network

Fig. 5. Input nodes positioning (fan-out based)

input nodes i and j. Larger number of common descendants with smaller distance to the parents pair result in larger proximity factor and therefore, coressponding nodes should be placed in a closer distance to each other. Proximity factors for different pairs of the sample DFG in Fig. 7 have been calculated as well.

Input nodes placement algorithm: A heuristic algorithm is introduced below which employs the proximity factor for positioning input nodes of a DFG onto the LSRDP input ports. Here are some definitions.

P: is the matrix of proximity factor for input node pairs.

$$P = \begin{bmatrix} \infty & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & \infty & \cdots & p_{2,n} \\ & \cdots & \cdots & \\ p_{n,1} & p_{n,2} & & \infty \end{bmatrix} \tag{2}$$

$$p_{i,j} = p_{j,i} = \begin{cases} \infty & if \quad i = j \\ \sum_{k \in S_{i,j}} \frac{1}{D_{k,i}} & if \quad i \neq j \end{cases} \tag{3}$$

L: is the input ports array which stores the list of placed nodes such that the indexes indicate the location of corresponding input node.

l and *r*: denote the index of candidate locations in *L* for placing the under process input node.

$C_{l,m}$ and $C_{r,m}$: show the amount of proximity of an under process node *m* to the previously placed input nodes which have been located between *l* and *r* in array *L*. The node *m* will be placed in location *l* if $C_{l,m}$ is larger than $C_{r,m}$ otherwise, it will be located in location *r*. $C_{r,m}$ and $C_{l,m}$ are calculated as:

$$C_{l,m} = \sum_{i=l+1}^{r-1} \left(p_{i,m} \times \frac{1}{|i-l|} \right) \quad (4), \quad C_{r,m} = \sum_{i=l+1}^{r-1} \left(p_{i,m} \times \frac{1}{|r-i|} \right) \quad (5)$$

In Eq. 4 and 5, $p_{i,j}$ is the proximity factor of node *m* and node *i* that has already been placed. The second term is the inverse of distance of candidate location (*l* or *r*) to the location of node *i*. In this way, the amount of proximity to already placed nodes is examined and one of the candidate locations (*l* or *r*) is chosen.

Placement alg.

```
1. Construct matrix P including the proximity factors for each pair of input nodes (n is the number of input nodes). Initialize $L = \Phi$ .
2. Find node m with the highest proximity factor from the first row of matrix P(i= 1) and place it in array L so that L[n/2]= m.
3. Initialize l and r to n/2-1 and n/2+1, respectively.
4. Find the next node (m) with the highest proximity factor from the first row of matrix P(i= 1).
5. Calculate $C_{l,m}$ and $C_{r,m}$ using Eq. 4 and Eq. 5.
6. if $C_{l,m} > C_{r,m}$:
       l= l+1, L[l]= m (node m is placed in the location l)
   else:
       r= r+1, L[r]= m (node m is placed in the location r)
7. if still there is any unplaced input node, go to step 4.
```

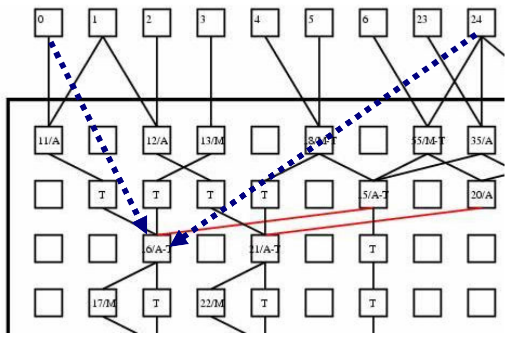
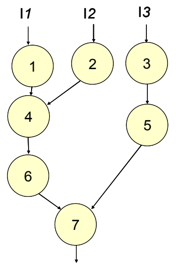


Fig. 6. A piece of mapping result for Poisson-3x3 DFG



$$S_{1,2} = S_{2,1} = \{4,6,7\}, p_{1,2} = p_{2,1} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = 1.08$$

$$S_{1,3} = S_{3,1} = \{7\}, p_{1,3} = p_{3,1} = 0.25 \,,$$

$$S_{2,3} = S_{3,2} = \{7\}, p_{2,3} = p_{3,2} = 0.25$$

$$S_{1,1} = S_{2,2} = S_{3,3} = \infty$$

Fig. 7. A sample DFG

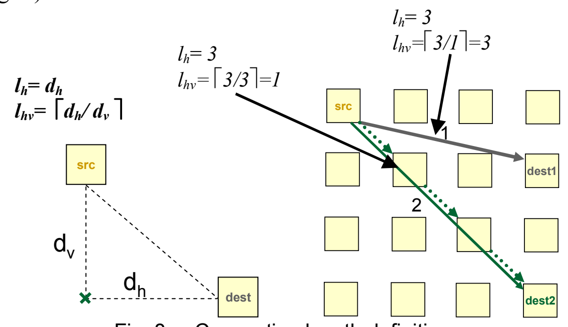
3-4. Connection-length minimization

In the basic placement algorithm, connection length between any source and destination nodes is evaluated as the horizontal distance of two nodes (connection length= l_h= d_h: horizontal distance). For two nodes located in two consecutive rows (d_v= *1*), the only possible way for routing is through the available ORN resources. On the other hand for two nodes placed in inconsecutive rows (vertical distance: d_v> *1*), it is possible to use intermediate TUs for routing. In this way, the connection can be segmented to d_v*-1* connections between consecutive rows.

Considering above point, we use an alternative measurement for the connection length as $l_{hv}=\lceil d_h/d_v \rceil$. In Fig. 8 (left-side) the two definitions of connection length have been displayed. In right-side two connections (denoted by 1 and 2) can be seen so that d_{h1}= d_{h2}= 3, while they have different vertical distances (d_{v1}= 1, d_{v2}= 3). For connection 1, connection length would be 3 and the only possible way for routing from *src* to *dest1* is via the ORN switches. On the contrary, for connection 2, it is possible to use available intermediate TUs and break it into three segmented connections from *src* to *dest2*. This results in reducing connection length to one.

By using above new definition for connection length measurement, the placement algorithm was modified such that the vertical connection length becomes effective in calculating the cost function. Fig. 9 shows an example of placing a node which has two parents (P1 and P2) already assigned to two PEs.

Obviously due to the routing resource constraints (particularly of being unidirectional) descendant nodes can be placed on the PEs of their parents' succeeding rows (indicated as candidate rows in Fig. 9).



$l_h = d_h$
$l_{hv} = \lceil d_h / d_v \rceil$
src
d_v
d_h
dest
$l_h = 3$
$l_{hv} = \lceil 3/3 \rceil = 1$
$l_h = 3$
$l_{hv} = \lceil 3/1 \rceil = 3$
src
1
dest1
2
dest2

Fig. 8. Connection length definitions

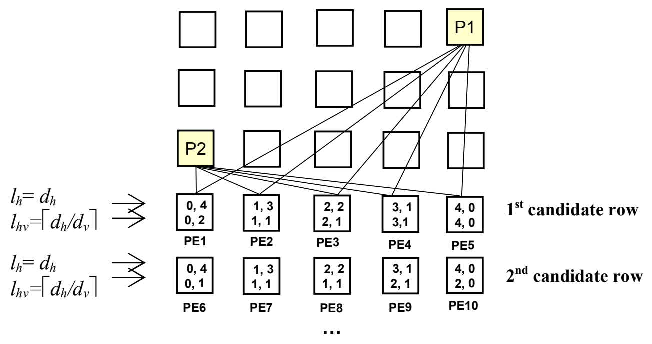


Fig. 9. Examining different PEs of LSRDP for placing a DFG node

The connection length has been calculated based on two abovementioned measurements for each candidate PE. By using lh for connection length measurement, PE2 is the best choice which gives the minimum MCL equal to two. In the second candidate row, choosing PE8 gives the same result, however PE2 is proffered due to smaller vertical distance. On the other hand, connection length measurement based on lhv results in choosing PE2 which gives the least value of MCL equal to one in the first row. Although PE6, PE7 and PE8 in the second row give the same result, but due to smaller vertical distance, PE2 is chosen eventually.

4. Experimental Results

We conducted experiments to evaluate different LSRDP architectures by means of the proposed mapping techniques. Input DFGs (Table 1) were extracted from the four target applications as mentioned in Section 3-1. Three PE structures (Fig. 1) and two LSRDP layouts (introduced in Section 2) were examined. Due to various number of inputs and outputs of the PE architectures, three ORN structures were designed and their area were calculated as in Table 2. In order to calculate PE area, it is assumed that the number of Josephson Juntions (JJs) required for implementing a TU or mux in a PE is almost %10 of a double-precision functional unit [4].

Table 3 shows results of evaluations of various LSRDP architectures. We used both techniques presented in Section 3-3 for placing input nodes. In addition the MCL minimization technique introduced in Section 3-4 were exploited to reduce the MCL size. In Table 3, S1 denotes first strategy including fan-out based input nodes placement and using l_h as the cost of connection lengths. S2 stands for a strategy employing the proximity-factor based input nodes placement and l_{hv} for the connection length measurement. Results of experiments show the effectiveness of the proposed techniques in reducing MCL, ORN size as well as the overall LSRDP size. It is observed from the implementation that each CB needs around 550JJs and each FU i.e. ADD/SUB or MUL requires around 40KJJs.

Table 1. specification of the extracted DFGs

	# of nodes	# of inputs	# of outputs	# of ops	max. inp. nodes fan-out	Max. fan-out
Heat-8x1	34	6	4	16	2	1
Heat-8x2	60	8	4	32	3	3
Heat-16x2	172	16	12	96	3	3
Poisson-3x3	62	18	1	33	3	2
Vibration-4x2	48	8	4	24	3	2
Vibration-8x2	136	16	12	72	4	4
ERI-1	20	8	3	9	3	1
ERI-2	76	16	9	51	3	3
ERI-3	89	14	9	66	3	6
ERI-4	67	19	1	47	4	3

Table 2. PE types and ORN area calculations

	# of inputs/ outputs	PE Area (x FU)		ORN Area (x CBs)
		Layout-I	Layout-II	
PE1	3/2	2.1	1.1	1.5xWx(4xMCL)
PE2	4/3	2.2	1.2	2xWx(6xMCL+2)
PE3	3/3	2.2	1.2	1.5xWx(4xMCL+1)

Table 3. Evaluation results for various architectures

		Layout-I		Layout-II	
		S1	**S2**	**S1**	**S2**
MCL	**PE basic arch.**	14	6	15	12
	PE arch. I	8	3	9	4
	PE arch. II	10	4	12	7
ORN size (overall) x CB	**PE basic arch**	25116	12600	29250	24336
	PE arch. I	30600	13680	38080	17680
	PE arch. II	18696	8237	23520	14790
No. of FUs (overall) x FU	**PE basic arch**	580	683	330	344
	PE arch. I	634	713	384	384
	PE arch. II	627	669	360	384
Overall LSRDP Area (JJ)	**PE basic arch**	**36923K**	**34193K**	**29200K**	**27040K**
	PE arch. I	**48083K**	**35995K**	**36190K**	**25031K**
	PE arch. II	**35307K**	**31258K**	**27266K**	**23451K**

The LSRDP area is composed of the area of ORNs and PEs. Area of ORNs has been represented by the number of cross-bar switches and the area of PEs is estimated by the number of required FUs. Overall area of the LSRDP (the last row) has been reported in terms of the number of Josephson Junctions (JJs). Since each PE in Layout-I implements both ADD/SUB and MUL operations, it needs larger overall area regardless of the strategy of mapping. Comparing PE architectures, although PE arch. I gives smaller MCL size, but it results in larger overall area for both layout types. As PE arch. I has four inputs and three outputs which are more than those of two other PE architectures, therefore, ORN size would be bigger referring to the information in Table 2. In Table 3 it can also be observed that the PE basic arch. and PE arch. II both obtain smaller overall area for LSRDP, however the MCL size for the first one is larger. Using mapping strategy S2, Layout-II and the PE arch. II the smallest area is achievable for the LSRDP.

5. Conclusion

A reconfigurable accelerator comprising of a large matrix of PE implemented by SDFG circuits is a key component for a computer featuring high performance and low power consumption. That is also suitable for executing massive computational-intensive scientific applications. A mapping tool has been developed in which the main goal is to reduce the LSRDP area. In future we intend to optimize the LSRDP architecture in attempt to decrease the overall area through exploring the design space. In the compiler side, the DFG generation from scientific applications will be performed automatically and the DFG customization for the underlying accelerator architecture will be developed.

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] Cell Broadband Engine, http://cell.scei.co.jp/index_j.html.
[2] ClearSpeed Processor, http://www.clearspeed.com/.

[3] De Micheli, G., Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994.

[4] I. 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.

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

[6] J. Makino, K. Hiraki and M. Inaba, GRAPE-DR: 2-Pflops massively-parallel computer with 512-core, 512-Gflops processor chips for scientific computing, SC07 2007.

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

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

[9] N. Sherwani, Algorithms for VLSI physical design automation, Kluwer-Academic Publishers, 1999.

[10] N. Takagi et al., "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.