An SIMD Architecture for Shortest-Path Search and Its FPGA Implementation

Yasuhiro Takei, Masanori Hariyama and Michitaka Kameyama
Graduate School of Information Sciences, Tohoku University
Aoba 6-6-05, Aramaki, Aoba, Sendai, Miyagi, 980-8579, Japan
Email: {takei, hariyama, kameyama}@ecei.tohoku.ac.jp

Abstract— *Shortest-path search over graphs plays an important role in various applications. However, shortest path algorithms such as the Dijkstra's algorithm include complex processings. It is difficult for accelerators with fixed-datapath such as GPUs to accelerate these algorithms efficiently. This paper presents an FPGA-based accelerator with a SIMD architecture for the shortest-paths algorithm. In the proposed architecture, operations in the Dijkstra's algorithm are done with a high degree of parallelism, and the memory usage is reduced by using a memory management scheme. According to the evaluation, the proposed architecture is able to deal with graphs with more than 800,000 nodes on the Altera Stratix V.*

Keywords: Dijkstra's algorithm, Single Instruction Multiple Data, FPGA

1. Introduction

Recently, there is a huge demand to find the shortest-path for large scale graphs in many applications such as traffic simulation, social networking services and bioinformatics. The shortest-path problem is mainly classified into two types: single-source shortest-path problem (SSSP) and all pair shortest-path problem (APSP). Dijkstra's algorithm [1] and Bellman-Ford algorithm [2] are used to solve the SSSP. Warshall-Floyd Algorithm [3] is used to solve the APSP.

To accelerate the processing of large-scale graphs, there have been many software-based studies in terms of improving a data structure and reducing a computational amount reduction. Moreover, GPU[4] and FPGA[5] are used to accelerate the Warshall-Floyd Algorithm. However, it is difficult to accelerate these algorithms efficiently since most of the shortest-path algorithms include serial and complex data-flows. In order to process the shortest-path problem for large scale graphs, PC clusters with many CPUs are often used [6] because of their large memory capacity. However, these computing systems need very large space and power consumption.

To solve this problem, some FPGA-based accelerators have been proposed. FPGAs can implement application-specific data-paths by reconfiguration after fabrication. Moreover, the power consumptions of FPGAs are less than one-tenth of those of CPUs and GPUs. Tommiska[7], Fernandez[8], and Sridharan [9] have designed the FPGA-based architecture for SSSP with the Dijkstra's algorithm. However, their work did not consider processing large-scale graphs.

This paper presents an FPGA-based accelerator for the Dijkstra's algorithm. In order to accelerate processing and memory access in the Dijkstra's algorithm, we design the SIMD (single instruction multiple data) architecture. We consider how to search the shortest path with a high degree of parallelism, and how to reduce the memory usage on a limited memory space. The proposed architecture is implemented on the FPGA board for evaluating the resource usage.

2. Dijkstra's algorithm and implementation of an FPGA

The Dijkstra's algorithm is one of the most popular algorithm to solve SSSP. Because it is easy to implement, this algorithm is used in various applications such as analysis of the internet, traffic simulation and so on. Let the node where we are starting with be called S. Let $d(y)$ be the distance from S to node y. The flow of the Dijkstra's algorithm is represented by the following steps.

Step1: Assign to every node a tentative distance: set it to zero for S, and to infinity for all other nodes. Mark all nodes *"unvisited"*.

Step2: Select the *unvisited* node which has the smallest tentative distance and make it the *"current node"*.

Step3: For the *current node*, consider all of *unvisited* neighbor nodes and update their tentative distance. If the *current node* is A , and one of the *unvisited* neighbor node is B, set the tentative distance of B ($td(B)$) to $\min(td(B), d(A) + l_{AB})$,where l_{AB} is the length of the edge between A and B. When considering all of *unvisited* neighbor nodes of the *current node*, mark the *current node "visited"*.

Step4: Until all nodes are marked *visited*, go back to Step2.

The processing time of the Dijkstra's algorithm depends on searching the minimum distance in Step2 and updating tentative distances in Step3. In these processings, there are many comparison operations on multiple node data. The SIMD architecture is suitable for processing in parallel.

To process the Dijkstra's algorithm, a memory space for tentative distances and paths is required. Since these data are read and updated frequently, on-chip memory on an FPGA is suitable. However, the capacity of the on-chip memory is small. The memory management is required for reducing the memory usage and the processing time. It is unnecessary to store the distance on the visited nodes and infinity. These unused data should be replaced or avoid storing in the memory module.

3. Architecture

Figure 1 shows the overall architecture of the proposed FPGA-based accelerator. This architecture consists of a SIMD module, a memory controller, a FIFO, an adder and a current node register. The SIMD module is used for searching the minimum distance and updating tentative distances. The memory controller and the FIFO are used for transferring the graph data from an external memory to the SIMD module. The current node register stores the current node number and the distance from the start to the current node.

Figure 2 shows the architecture of the SIMD module. This module consists of block RAMs, comparators, and a counter-based address generation unit(AGU). The block RAMs store the values of node number, the tentative distance and the previous node. In the initial state, the block RAMs are empty.

For updating the tentative distances, the neighbor node number is searched in parallel as shown in Fig.3. Then the tentative distance at the neighbor node is compared with the sum of the distance at the current node register and the length of the edge. The tentative distance and the previous node are updated as shown in Fig.4. If the neighbor node number is not in the block RAMs, the data of the neighbor node number, the sum of the distance and the length, and the current node number are stored as new data.

For the searching of the minimum distance, comparators are connected as shown in Fig.5. When the minimum value searching is completed, the value of the minimum distance and the node number are stored in the current node register. The memory space for the current node can be overwritten as shown in Fig.6. Hence, the memory usage for the tentative distance can be reduced.

Let us consider implementing the graph data on the FPGA. In related works of the FPGA implementation of the Dijkstra's algorithm [7],[8],[9], the adjacency matrix is used because the overhead of the memory access is small. However, very large memory space is required for unnecessary data that indicates unconnected edges if the graph is sparse. In this work, adjacency list is used for using a limited memory space in FPGA efficiency. The lengths of edges and the neighbor node number are stored in the external memory since the amount of these data is large. By using a index pointer in the memory controller, the length of the edge and the neighbor node number are transferred from the external memory to the FPGA.

4. Evaluation of the proposed architecture

We use the Terasic DE5-NET FPGA board [10]. This board includes an Altera StratixV 5SGXEA7N2F45C2, and a DDR3 SDRAM (4GB). Altera Quartus 13.1 is used for design. For a proto-type design, we implement the SIMD module for shortest-path search in 32 nodes. Table 1 shows the resource usage. The resource usage changes by the numbers of block memories and comparators. The degree of parallelism can increase if many block memories and comparators are implemented. However, the logic utilization becomes large.

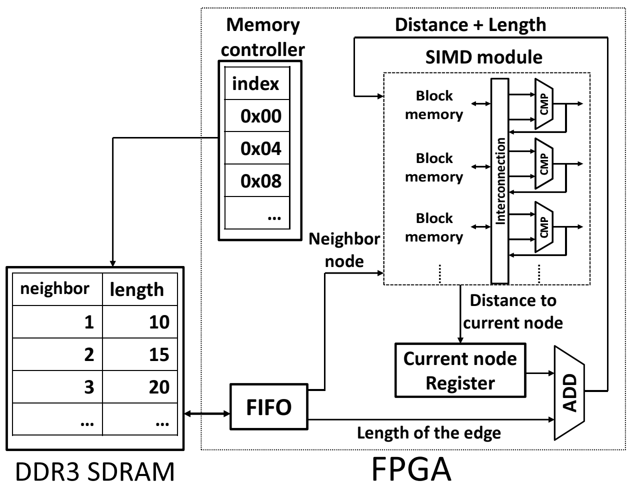


Fig. 1: Overall architecture

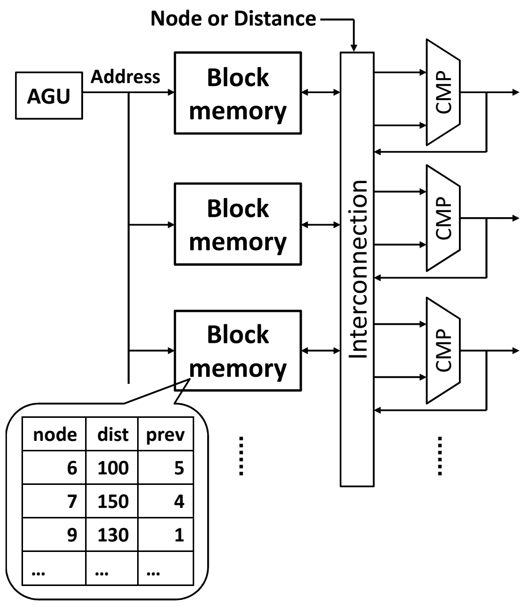


Fig. 2: SIMD module

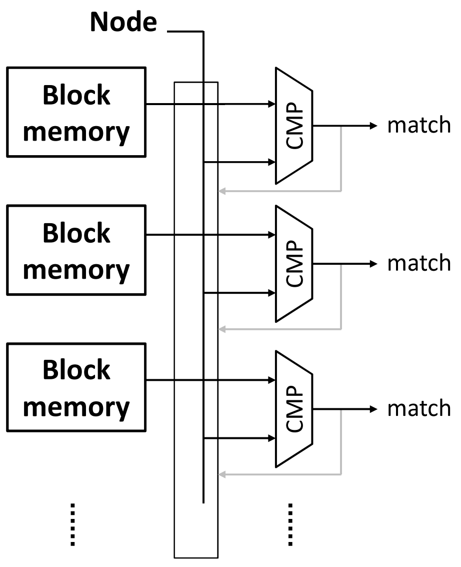


Fig. 3: Searching a node number in the SIMD module

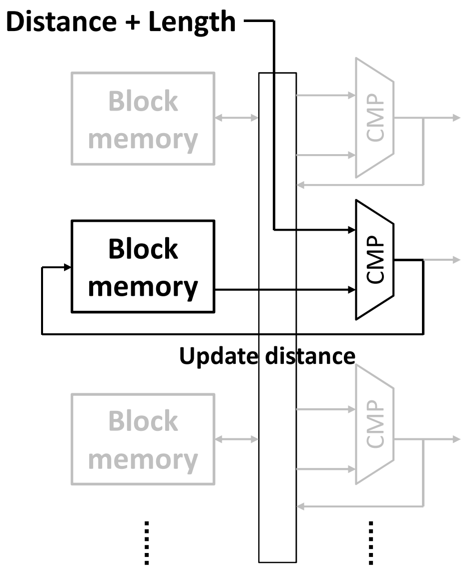


Fig. 4: Updating data in the SIMD module

Note that the number of block memory module on the FPGA is 5120.

Let us consider the memory usage for storing the tentative distances and node numbers. The FPGA has about 50M bits of on-chip memory. Let the number of bits in every node be 64, about 800,000 nodes can be stored in the FPGA. According to Section 3, the data of the visited nodes can be replaced by the data of the unvisited nodes, and a memory space is not required if the tentative distance is infinity. Hence, the proposed architecture can process the shortest path search on very large scale graphs with more than 800,000 nodes.

The processing time depends on the amount of data in the block memories. If the input graph is sparse, the amount of data in the block memories is small, and the processing time is small. Most of the large scale graphs in the real world, such as road network, traffic network, social network and so on, are sparse. Hence, the proposed architecture may be suitable for processing the large scale graphs in the real world. We are now designing the overall architecture as shown in

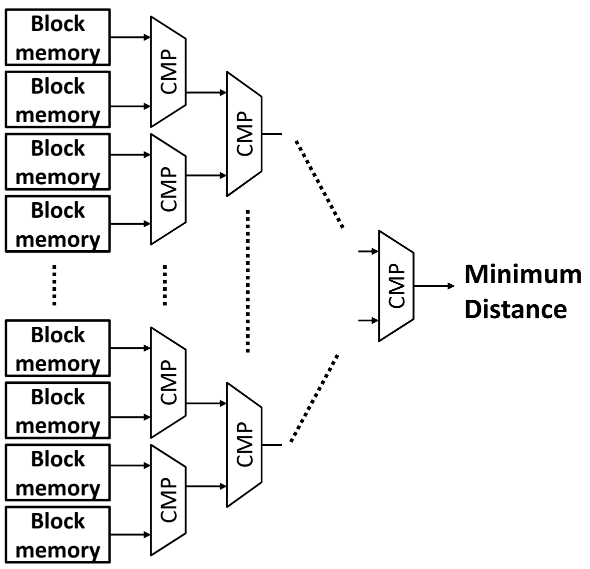
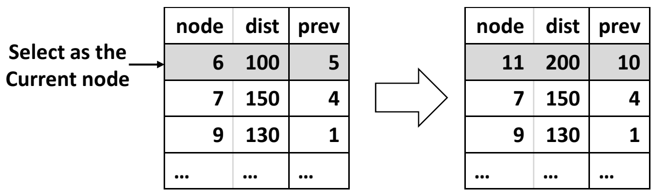


Fig. 5: Searching the minimum distance in the SIMD module



node	dist	prev
6	100	5
7	150	4
9	130	1
…	…	…

node	dist	prev
11	200	10
7	150	4
9	130	1
…	…	…

Fig. 6: Overwriting unused data in the block memory

Fig.1, and measuring the total processing time of the Dijkstra's algorithm.

5. Conclusions

We have proposed an FPGA-based accelerator with an SIMD module for a shortest-path search. We discussed about how to parallelize the Dijkstra's algorithm and how to use limited memory space on FPGA boards. According to the evaluation, the proposed architecture can accelerate shortest-path search on large scale graphs with more than 800,000 nodes on the Altera Stratix V.

In future works, we are going to implement large scale architecture on the FPGA board in order to accelerate applications with shortest-path problems such as the traffic network analysis. Moreover, it is very interesting to implement improved shortest-path algorithms, such as the A* algorithm[11] and the high-way dimension algorithm[12] on an FPGA.

Table 1: Resource usage

Block memory	Comparator	LUT	Register	Memory bit
2	8	282	152	512
4	4	137	68	512
2	16	572	321	1024
4	8	282	382	1024

Acknowledgement

This work is supported by JSPS KAKENHI grant number 24300013.

References

[1] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", Numerische Mathematik, 1(1): pp.269–271, 1959.
[2] R. Bellman, "On a Routing Problem", Technical report, DTIC Document, 1956.
[3] R. W. Floyd, "Algorithm 97: Shortest Path", Commun. ACM, 5(6) pp.345–346, June 1962.
[4] G. J. Katz and J. T. Kider Jr, "All-Pairs Shortest-Paths for Large Graphs on the GPU", In Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware, 2008.
[5] U. Bondhugula, A. Devulapalli, J. Fernando, P. Wyckoff, and P. Sadayappan, "Parallel FPGA-Based All-Pairs Shortest-Paths in a Directed Graph", In Proceedings of Parallel and Distributed Processing Symposium, 2006.
[6] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. "Pregel: a System for Large-Scale Graph Processing", In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 135–146, 2010.
[7] M. Tommiska and J.Skytta. "Dijkstra's Shortest Paths Algorithm in Reconfigurable Hardware", In Proc. Field Programmable Logic and Applications, pp. 653–657, 2001.
[8] I. Fernandez, J. Castillo, C. Pedraza, C. Sanchez, and J. I. Martinez, "Parallel Implementation of the Shortest Path Algorithm on FPGA" In Proc. 4th Southern Conf. on Programmable Logic., pp. 245–248, 2008.
[9] K. S. T.K.Priya and P. Kumar, "Hardware Architecture for Finding Shortest Paths", In Proc. IEEE Region 10 Conf., pp. 1–5, 2009.
[10] Terasic, "DE5-NET FPGA Development Kit", http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=EnglishCategoryNo=158No=526.
[11] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", Systems Science and Cybernetics, IEEE Transactions on, 4(2):pp.100–107, 1968.
[12] I. Abraham, A. Fiat, A. V. Goldberg, and R. F. Werneck, " Highway Dimension, Shortest Paths, and Provably Efficient Algorithms", Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 782–793, 2010.