



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

Verilog Implementation of a System for Finding Shortest Path by Using Bellman Ford Algorithm

¹Mitesh Bisen, ²Vijayshri Chaurasia, ³Ajay Somkuwar
Department of Electronics and Communication Engineering
Maulana Azad National Institute of Technology, Bhopal
INDIA

Abstract: There exist several high speed computation applications that require minimum distance computation between nodes in a weighted graph. Dedicated techniques exist for this purpose and are called as shortest path computation. Depending upon the need of application, there are three types of shortest path problems; Single Source Shortest Path (SSSP), Single Destination Shortest Path (SDSP) and All Pair Shortest Path (APSP). For Single source shortest path computation Bellman Ford Algorithm provides most efficient results. This paper presents the Verilog implementation of Bellman Ford algorithm for determination of shortest path between one node to all other nodes of a graph. This implementation leads to the hardware implementation of Bellman Ford algorithm which is not possible with existing implementation.

Keywords: Bellman Ford, single source shortest path, source node, destination node.

Introduction

If any path has minimum weight between sources to destination among existing several is called shortest path between two nodes. Shortest path problem with real weights is a problem of finding shortest path between specified node known source node and destination node. Shortest path problem studied since 1958 [5], and it has applications in various fields. The problem has wide application in road network, chip designing, robot navigation, urban traffic problem, approximating piecewise linear function etc [10-11]. Various shortest path algorithms have been studied in [4-7]. Optimal shortest path algorithms for general network topologies are the Dijkstra's [6] and the Bellman-Ford algorithm [5]. Bellman Ford algorithm that computes shortest path between given source to all possible destinations in a weighted graph, it is slower than Dijkstra for same application since it computes weight for all destinations whereas Dijkstra computes for specified two nodes. But Bellman Ford algorithm is more versatile than Dijkstra as it is capable to deal with negative weight edges. Negative edges are found in various applications hence it has importance [7]. Another important thing about BFA is it detects negative cycle in graph also [14]. The Bellman-Ford algorithm can be efficiently implemented in a distributed computational environment while the Dijkstra's algorithm is inherently sequential [1, 4]. The distributed version of the Bellman Ford algorithm can solve an SSSP problem in $O(n)$ time units where n is graph and n is no. of nodes present in graph. However, the distributed algorithms for general network topologies need much more time complexities than the algorithms designed for a special class of networks, since a substantial amount of time is needed to deal with the possible worst cases which rarely occur. One way to obtain better time complexities is to design a special class of networks and the associated algorithms. In [4], a shortest path algorithm for the hierarchically clustered networks is proposed. The algorithm has the computation time complexity of $O(VE)$ where V are vertices and E are edges of graph, for solving an SSSP problem. The communication time complexity of $O(VE)$, implicitly assumes that it can take $O(1)$ time units to make $O(i)$ initializations for the broadcasting. However, due to the broadcasting procedure nested in the main iteration loop, the communication takes $O(\log_2(n))$ time units if we assume that it takes $O(i)$ time units to make $O(i)$ initializations for the broadcasting, i denotes iteration. BFA is the most suitable for solving SSSP problem in any weighted graph. This paper gives the Verilog implementation of BFA. In rest of the paper section II explains BFA. In section III methodology of SSSP computation using BFA is explained. Verilog implementations of BFA along with results obtained are presented in section IV. Finally conclusions are made in section V.

I. Bellman Ford Algorithm

Like Dijkstra's Algorithm, Bellman-Ford is based on the principle of relaxation, in which an approximation to the correct distance is gradually replaced by more accurate values until eventually reaching the optimum solution. In both algorithms, the approximate distance to each vertex is always an overestimate of the true distance, and is replaced by the minimum of its old value with the length of a newly-found path. However, Dijkstra's algorithm greedily selects the minimum-weight node that has not yet been processed, and performs this relaxation process on all of its outgoing edges; in contrast, the Bellman-Ford algorithm simply relaxes all the edges, and does this $|V| - 1$ times, where $|V|$ is the number of vertices in the graph. In each of these

repetitions, the number of vertices with correctly-calculated distances grows, from which it follows that eventually all vertices will have their correct distances. This method allows the Bellman–Ford algorithm to be applied to a wider class of inputs than Dijkstra algorithm. BFA runs in $O(|V| \cdot |E|)$ time, where $|V|$ and $|E|$ are the number of vertices and edges respectively. When the algorithm is used to find shortest paths, the existence of negative cycles is a problem, preventing the algorithm from finding a correct answer. However, since it terminates upon finding a negative cycle, the BFA can be used for applications in which this is the target to be sought; for example in cycle-cancelling techniques in network flow analysis[12-15]. Bellman ford algorithm also useful in determining the all pair shortest path but, it is needed to run $n - 1$ time separately.

III Methodology

The variable D_i stands for the sought minimal distance from node 1 to the i^{th} node. The solution goes through stages with and every stage includes a new node and the new value for D_i is recalculated. This goes on until the resulting values can no longer get smaller. Variable k stands for the number of the stage. In the initial stage $k = 1$ all D_i are equal to the direct distance from node 1 to the corresponding node. R_{1i} denotes weight between node 1 and i^{th} node [14-15].

$k = 1, D_i = R_{1i}, i = 1, 2 \dots n.$

In first stage $k = 1, D_i = R_{1i}$ it symbolize direct Distance between node 1 and node i , if there is no arch i to j , then it will be replaces by my maximum weight says infinite.

$k = 2, D_1 = 0$, since node 1 to node 1 there is no self loop it takes zero value correspond to it, in second iteration or stage to D_i will be replaces by its pervious value and new value of it if possible smaller compare to old will be get in stage to operation

$$D_2 = \min\{D_2, D_1 + R_{12}, D_3 + R_{32}, \dots, D_n + R_{n2}\} \quad (1)$$

$$D_3 = \min\{D_3, D_1 + R_{13}, D_2 + R_{23} \dots D_n + R_{n3}\} \quad (2)$$

$$D_n = \min\{D_n, D_1 + R_{1n}, D_2 + R_{2n} \dots D_{n-1} + R_{n-1n}\} \quad (3)$$

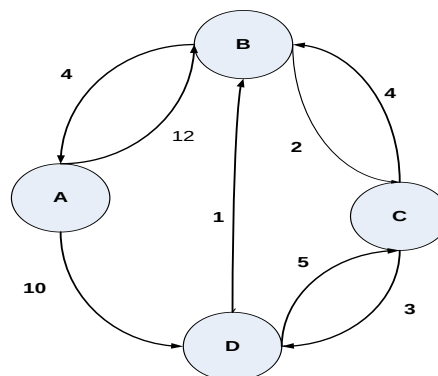
$k = 3, 4 \dots n - 2.$

This procedure is repeated until we get two identical rows of distance in the table or until $n - 2$ is reached in the worst case.

II. Implementation and result

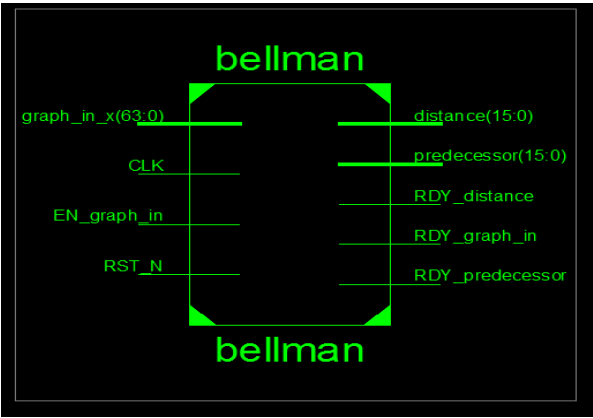
The Bellman Ford algorithm is implemented in Verilog by using Xilinx-13.2. By using this we have to find single source shortest path in any given graph having 'n' no. of nodes. Figure I represent a graph having four nodes A, B, C, D. Where A is source node and B, C, D is destination nodes.

Figure I 4 node graph taken for testing Bellman Ford algorithm



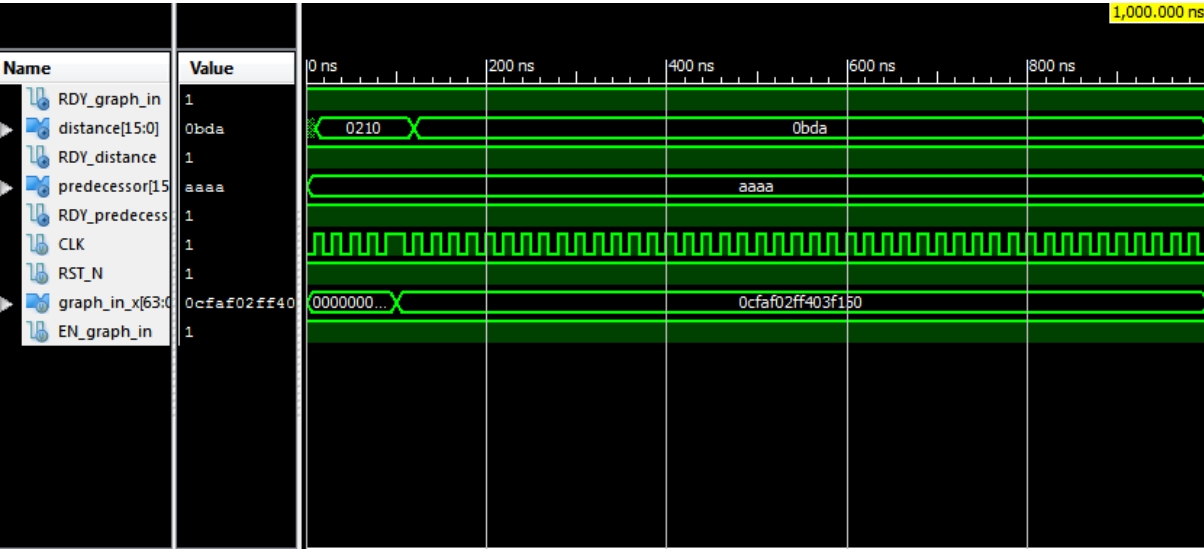
In figure II we have the 9 pin diagram which is generated after simulation of Bellman Ford algorithm in verilog. Here graph_in_(63:0) is used for input and input is given in matrix form having 64 bits. CLK is there for giving clock for the ckt. EN_graph_in is to enable the input. RST is used to reset the previous result. The output will be taken from pin distance (15:0). Output is of 16 bits. In the code each element of 4 by 4 matrix represented by 4 bit binary no. for 16 element there is 64 bit input at pin graph_in, output will be of 16 bit first 4 bit will denote distance between node A to A next 4 bit will denote distance A to B, next 4 bit similarly distance between A to C and A to D.

Figure II Top level schematic of Bellman Ford algorithm



Here the generated output waveform showing the output of each pin of pin diagram shown in figure II. Initially no input was there. After 400ns when input is applied in pin graph_in then corresponding output with clock will be generated, this is shown in figure III. We got the SSSP from A to A, B, C and D as ‘**0bda**’ i.e. $A \rightarrow A$ as 0, $A \rightarrow B$ as 11, $A \rightarrow C$ as 13 and $A \rightarrow D$ as 10.

Figure III Input-output waveform by simulation in verilog



III. Timing detail

While finding shortest path the timing is main aspect. Our aim is always to reduce the total time. The time taken is directly proportional to the total no. of nodes there in the graph.

Here we have taken 4 nodes graph and the timing detail of that graph is given below

All values displayed in nanoseconds (ns)

Timing constraint: Default OFFSET IN BEFORE for Clock 'CLK'

Total number of paths / destination ports: 696 / 9

Offset: 7.446ns (Levels of Logic = 6)

In the graph we have to move from source to destination. Here the source is ‘graph_in’ and the destination is ‘distance_6’. The total time is taken in the process is 7.446ns.

Here the table given below defines the separate time consume by each element and then total time.

Some Look Up Tables are generated in the architecture the processing time of each LUT is given in the table

Table I Timing detail

Cell:in->out	Gate fanout	Net Delay	Delay	Logical Name (Net Name)
IBUF:I->O	2	1.218	0.622	graph_in_x_50_IBUF (graph_in_x_50_IBUF)
LUT4:I0->O	1	0.704	0.424	distance_cmp_eq00011_SW0 (N5)
LUT4:I3->O	2	0.704	0.482	distance_cmp_eq00011 (N1)
LUT4:I2->O	1	0.704	0.000	distance_cmp_eq0001_wg_lut<4> (distance_cmp_eq0001_wg_lut<4>)
MUXCY:S->O	5	0.864	0.712	distance_cmp_eq0001_wg_cy<4> (distance_cmp_eq0001)
LUT2:I1->O	1	0.704	0.000	distance_mux0000<9>1 (distance_mux0000<9>)
FD:D		0.308		distance_6
Total		7.446ns	(5.206ns logic, 2.240ns route)	(69.9% logic, 30.1% route)

Total time require to generate output is 7.446ns, time taken to compute logic is 5.206ns which is about 69.9%,and in routing 2.240ns which is about 30.1%.

The table II presents the availability and utility of slices, look-up table (LUT), input output buses (IOBs) and clock. The proposed design results 1% utilization of slices, 8% utilization of 4-input LUTs and IOBs and 100% utilization of GCLKs.

Table II Device utilization summary

Logic utilization	Used	Available	Utilization
Number of slices	35	9312	1%
Number of 4 input LUTs	780	9312	8%
Number of bonded IOBs	411	4656	8%
Number of GCLKs	411	411	100%

IV. Conclusion

Bellman Ford algorithm is faster than other known algorithm for finding single source shortest path. Floyd warshall algorithm is useful in all pair shortest path computation but it will take more time for SSSP because it will require running separately to other node and time taken will be large. Similarly Dijkstra's is more efficient for two node shortest path application but SSSP application, it consumes more time due to $n - 1$ times initialization for n node graph. In present edge of science and technology it is necessary to have hardware implementation of any technique for various applications. In this paper we have presented verilog implementation of Bellman Ford algorithm which can be used further to generate using Field Programmable Gate Array. FPGA based hardware is faster and its utility is more as compared to general purpose processor. The speed up offered by the design may further be improved by adopting interconnection topology that maximize data transfer rate among processor element.

References

- [1] D. Bertsekas and J. Tsitsiklis, Parallel and Distributed Computation, Numerical Methods, Prentice Hall, Englewood Cliffs, New Jersey, 1989.
- [2] G. Huang and Shan Zhu, "A fast distributed optimal routing algorithm for multi commodity large data networks", The 33rd IEEE Conference on Decision and Control, Orlando, Florida, 1994.
- [3] J. K. Antonio, G. Huang and W. K. Tsai, "A fast distributed shortest path algorithm for a class of hierarchically clustered data network," IEEE Transactions on Computers, Vol. 41, No. 6, pp. 710-724, 1992.
- [4] G. B. Dantzig, "On the shortest route through a network," MAA Studies in Mathematics, vol. 11, part 1, pp. 89-93, The Mathematical Association of America, 1975.
- [5] E. W. Dijkstra, "A note on two problems in connection with graphs," Numerische Mathematic, Vol. 1, pp. 269-271, 1959.
- [6] R. Bellman, Dynamic Programming. Princeton University Press, Princeton, New Jersey, 1957.
- [7] Weisstein, Eric. "Floyd-Warshall algorithm". Wolfram math world.
- [8] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein introduction to algorithms, 3rd ed. Massachusetts, USA: the MIT Press 2009.
- [9] SEDGEWICK, -H., Alprithins (Addison- Wesley, Reading, MA, 1988 2nd edn.)
- [10] Huelsbergen, L. "A representation for dynamic graphs in reconfigurable hardware and its application to fundamental graph algorithm" Proc. ACM int Symp on field Programmable Gate Array 2000, pp 105-115
- [11] Bang-Jensen, Joegen, Gutin, Gregory, Section 2.3.4: "The Bellman-ford-Moore algorithm", 2000.
- [12] "Lecture 12: Shortest paths". Network Flows and Graphs. Department of Industrial Engineering and Operations Research, University of California, Berkeley. 7 October 2008.
- [13] K. Matsumoto and S. G. Sedukhin, "The Algebraic Path Problem on the Cell/B.E. Processor," The University of Aizu, Tech. Rep. 2010-002, 2010.
- [14] Bellman, Richard. "On a routing problem". *Quarterly of Applied Mathematics* **16**: 87-90. MR 0102435, 1958.
- [15] Ford, L.R. "Network flow theory Rand Corp., Santa Monica, CA P-923 1956.