International Journal of Advance Engineering and Research Development (IJAERD)
Volume 1,Issue 6,June 2014, e-ISSN: 2348 - 4470 , print-ISSN:2348-6406

Shortest Path Computation Using Minimum Spanning Tree in FPGA

Korrapati.Eswari Pujitha[1], Patale Vijaykumar[2], Nitin Sihna[3]

[1]*M.Tech Embedded System SENSE Department, VIT University, eswaripujitha@gmail.com*

[2]*M.Tech Embedded System SENSE Department, VIT University, patale.vijaykumar2013@vit.ac.in*

[3]*M.Tech Embedded System SENSE Department, VIT University, nitin03ece@outlook.com*

Abstract –The need for speedy execution is mounting in every field and almost all the application. Hence this utmost requirement in fields like routing wireless sensor nodes, Transportation network, mobile networks, etc can be quenched by the shortest path Computation. Because of the advantages of FPGA in terms of low cost, high computational power, they are used for calculating shortest path between several entities nowadays. The shortest path computation finds its relevance in the utility of minimum spanning tree algorithmswhichcomprises vagaries of renowned algorithms.The presented work accosts the use of prim and krushkal algorithm which prevails the greedy approach to find the desired result.The design envisages the performance of the system with consideration of 3, 6, 10, 20 and 40 nodes in Xilinx software.Along with that the power estimation and area occupancyis procured in cadence tool. Moreover, the hardware implementation has been carried out in Spartan3E.

Keywords—Shortest path, Minimum Spanning Tree , Prim Algorithm, Krushkal Algorithm, Spartan 3E, Cadence software.

I. INTRODUCTION

The road networks or mobile networks where multiple destination are to be covered. In road networks the destinations may be cities or towns or even villages. In telephone network the source may be users who are sending a message or doing a call to another user. The user receiving it is the destination. A single user can call any number of users in the network so it is a single source multiple destination scenario. To accomplish this need there should be a path connecting the source user and the multiple destination i.e all other users in the network. This path should be optimised to the most possible extent. This optimised path which cannot be further reduced is called shortest path of the network. The shortest path can be calculated by using many algorithms like Dijistra Algorithm, Minimum spanning tree algorithm, Bellman Ford algorithms etc.

In any network the source and the destinations are called as nodes of the network. All the possible links connecting the source and destinations are called paths. All the above mentioned algorithms can be implemented on a graph. A graph is a representation of vertices, edgesand their interconnections. It can be mathematically represented as G(V,E). To convert the above scenario into graph, represent the nodes as the vertices of the graph and paths as edges of the graph. Now this graph is called as connected graph .Graphs can be either directed graph or undirected graph.In directed graph the paths in the graph travel in specified direction. The directions of these paths cannot be reversed. In undirected graph the paths in the graph don't have specified direction; they can travel in any direction. In this scenario consider the undirected graph. If weights are assigned to the edges of the graph then the graph is called weighted graph. Now consider the time taken to travel between the two nodes or distances between the two nodes as the weight of the path connecting the nodes. In finding the shortest path for the entire graph, the paths with least weights are considered.

II. RELATED WORK

There is numerous numbers of shortest paths solving algorithms with single source multiple destinations scenario. There are different methods used for shortest path computation and they can also be compared [7].The shortest path algorithms have great demand as they reduce the travel path and the execution time[5].The FPGA implementation has many advantages like high computation

speed, low power consumption. FPGAs can also be reprogrammed as many times as possible. FPGAs are optimum to compute shortest path compared to general purpose processors which are very costly and comparatively of lesser speed. FPGA offers greater amount of flexibility by allowing it reprogram. The only limitation of FPGAs is they cannot be directly implemented on network with large number of nodes. In practical scenario the number of nodes in the network are very large in range of thousands .The FPGAs can be used here only when the network is scaled down to series of network with small nodes [13].Dijistra Algorithm is one of the algorithm with greedy problem solving technique. In this the paths or links connected should always be updated.This may require more resources i.e, more look up tables.Hence implementation of shortest path computation using dijistra may result to be costly than prim and krushkal [5].The prim algorithm and krushkal algorithm have their own pros and cons.Normally in network with large number of edges connected between the node like a real time mobile network scenario,prim algorithm has more advantages than krushkal algorithm. In network where large number of nodes are connected with single edges or less number of edges, krushkal algorithm works better than prim[6].

Further part of the paper is organised by discussing proposed model in third section, Simulation and synthesisresults in fourth section.Hardware Implementation of the proposed model is discussed in fifth section, followed by results and conclusion.

III. PROPOSED MODEL

The shortest path computation can be done by using many algorithms .The minimum spanning tree algorithms can be used for this purpose. The minimum spanning tree algorithms are optimal for finding the shortest path algorithm as they involve two algorithms which can deal with any scenario like network with large number of nodes or network with large number of edges.The minimum spanning tree algorithms can again be of two types. They are Prim Algorithm and Krushkal Algorithm.

A. *Prim Algorithm*

Prim algorithm is one of the most efficient algorithms. In Prim algorithm among vertices and edges, vertices play a major role. In single source multi destinations scenario, choose one of the vertex in the graph as source node and the remaining vertices arethe multiple destinations it has to meet. Let the graph be as shown in fig1(a).In PrimAlgorithm start from the source node(a). Look at the paths that directly connect the source node with the others nodes in the fig1(b). The weights of these paths should be taken into consideration. The path that has minimum weight should be included in minimum spanning tree let it be (x1) as shown in fig1(c). The sub tree starts from the source to the path that is selected. Then by considering this sub tree look for the paths that are connected to this sub tree. Among this paths select the path with lowest weight and include this path in the sub tree as shown in fig1(d). Now considering this sub tree continue the process of selecting the least weighted path and adding it to sub tree as in fig1(e). This is done till all the vertices in the graph are covered.Care should be taken such that adding a path into the subtree doesn't create a loop in the sub tree.After covering all the vertices the sub tree formed is nothing but our minimum spanning tree.The sum of weights of all the paths in the tree is the weight of the shortest path.

1) Prim algorithm example:

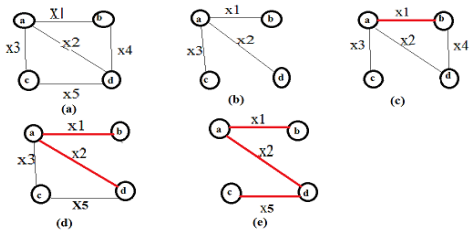


Fig.1 Graph with four nodes.

2) Pseudo code:

//Input :A weighted graph G with vertex (V),and edges (E)
//Output:The tree of the graph with minimum weight
Vt←V U{v1}
Et←null
For all i←1 to |V| -1 do
To find the spanning tree with minimum weight do e*=(v*,u*) from the edges present in the entire graph so that v is in V and u is present in V –Vt
Vt←Vt U {u*)
Et←Et U {e*}
The time complexity of the Prim algorithm is represented by Big O. So the time complexity of Prim algorithm is O(V2).

B.Krushkal algorithm

The other type of minimum spanning tree algorithm is Krushkal algorithm .In Krushkal algorithm all the paths that connect the nodes i.e all the edges in the graph should be considered . In Krushkal algorithm, the edges are givenmostimportance than vertices. From the basic fig.2(a), edges are selected based on their ascending order fig.2(b) describes the selection of the least weight edge(x5).Similarly the proceeding least weighted edges are routed to complete the minimum spanning tree as shown in fig2(c)&(d).Care should be taken such that the sub graph doesn't form a loop. This procedure is continued till all the vertices in the graph are connected with the sub graph which forms the minimum spanning tree as shown in fig2(e). The weight of this minimum spanning tree is the shortest path connecting the single source and multiple destinations.

1) *krushkal algorithm example:*

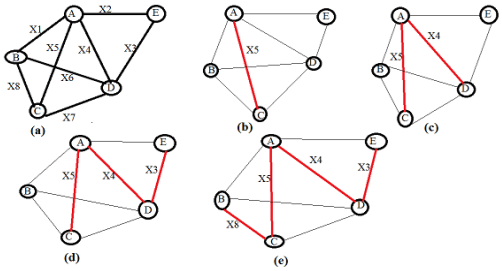
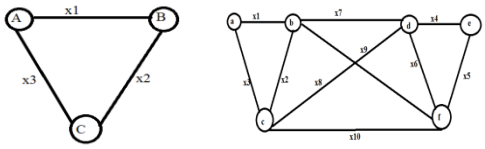


Fig.2 Graph with five nodes

2) *Pseudo code*:

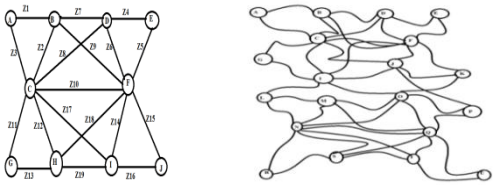
//Input: A weighted graph H=(V,E)
//Output: Shortest path of the graph
Et ←null; total←0//the starting values are considered.
i←0
while total < |V| -1
do
i← i +1
if Et U {e*} is acyclic
Et ← Et U {e*}; total ← total +1
Return Et

A network can be of any number of nodes. In real time the network will be having large number of nodes. The graph with three, six,ten, twenty and forty nodes is shown in fig.3(a)&3(b), fig.4(a)&4(b),fig5.



(a) (b).

Fig.3Graph with 3nodes and 6 nodes.



(a) (b)

Fig.4 Graph with 10 nodes and 20 nodes.

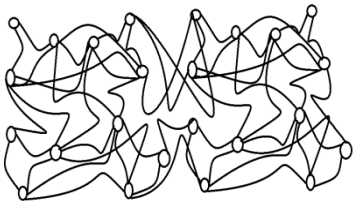


Fig.5 Graph with 40 nodes

IV. SIMULATION AND SYNTHESIS RESULTS

In the following approach, which exploits the flexibility of verilog code, the synthesis and simulation was targeted at Xilinx 12.2 Spartan3E family on XC3S250E device with package TQ144 at -4 grading speed. Verilog code of prim algorithm for graphs with 3,6,10,20,40 nodes as shown in fig.2,3,4 as written and synthesised .The resource requirement for these graphs like no of LUTs, adders, comparators they use is acquired and tabulated in table1.

TABLE1.

Resources required for prim algorithm in computing shortest path for 3,6,10,20,40 nodes

No of nodes	No of LUTS	Comparators	Adders
3	42	13	1
6	50	38	6
10	60	70	13
20	156	152	29
40	310	328	59

.

The resources required can also be plotted .This is represented in below Fig6.

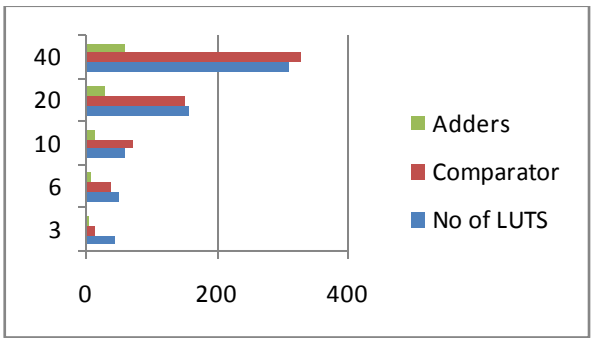


Fig6..Plot of resources required for Prim Algorithm in computing shortest path for 3,6,10,20,40 nodes.

The resources required by the graph increases as the number of nodes in the fig6 increases. The increase in the resources required is mostly linear. Verilog code of prim algorithm for graphs with 3,6,10,20,40 nodes as shown in fig.2,3,4 as written and synthesised .The resource requirement for these graphs like no of LUTs, adders, comparators they use is acquired and tabulated in table2.

TABLE2.
Resources required for krushkal algorithm in computing shortest path for 3,6,10,20,40 nodes

No of nodes	No of LUTS	Comparators	Adders
3	19	13	1
6	44	14	4
10	88	22	9
20	156	152	28
40	310	328	59

The resources required can also be plotted .This is represented in below Fig 7. The resources required by the graph increases as the number of nodes in the graph increases. The increase in the resources required is mostly linear.

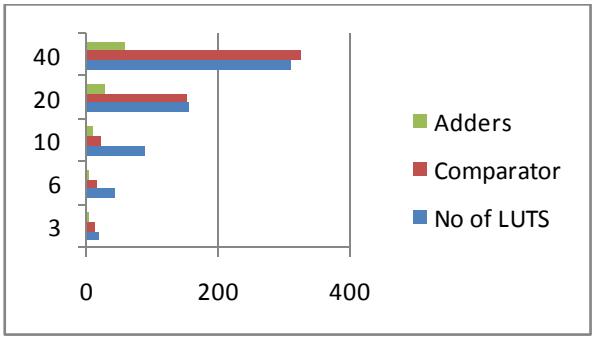


Fig.7Plot of resources required for krushkal Algorithm in computing shortest path for 3,6,10,20,40 nodes

The schematic diagram of the verilog code can be viewed in cadence. The RTL schematic of the krushkal algorithm for the graph with 6 nodes is shown in fig 8.The verilog code for 6 node graph can be written by creating two sub modules of 3 node graph and 1 comparator. The schematics for the comparator and 3 node graph which are used in finding the shortest path for 6 node graph are shown in fig 9& fig 10.

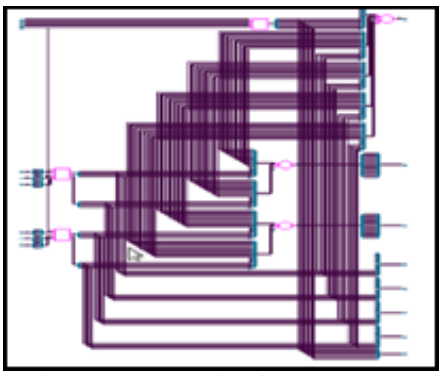


Fig.8 RTL schematic of 6 node graphfor finding shortest path using krushkal

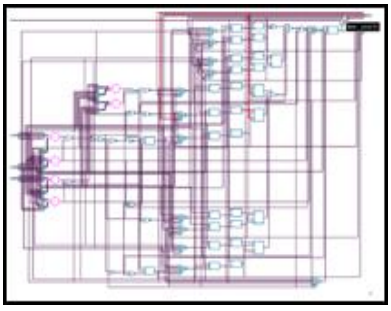


Fig.9 RTL schematic of Comparator used in 6 node graph

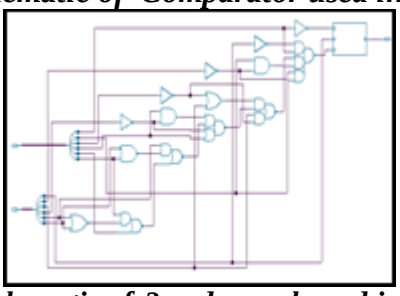


Fig.10 RTL schematic of 3 node graph used in 6 node graph

A. SIMULATION

The simulation for verilog code of prim algorithm can be simulated on either model sim or cadence tool. Simulation results of graphs with 3,4,6,10,20,40 nodes using prim algorithm can be shown in below fig.8,9,10,11,12,13

Fig.11 Simulation results of 3 node graph using modelsim

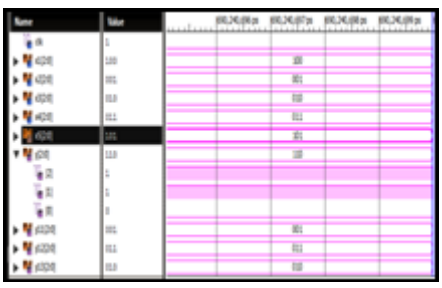


Fig.12 Simulation results of 4 node graph using modelsim

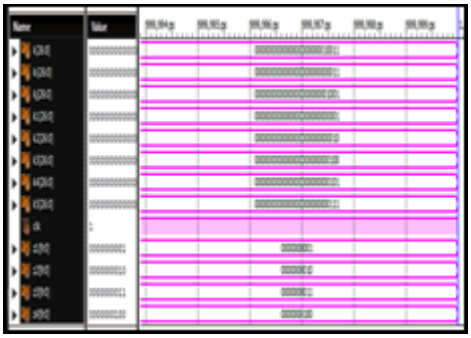


Fig.13 Simulation results of 6 node graph using modelsim

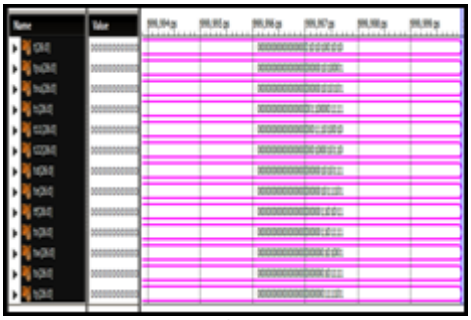


Fig.14 Simulation results of 10 node graph using modelsim

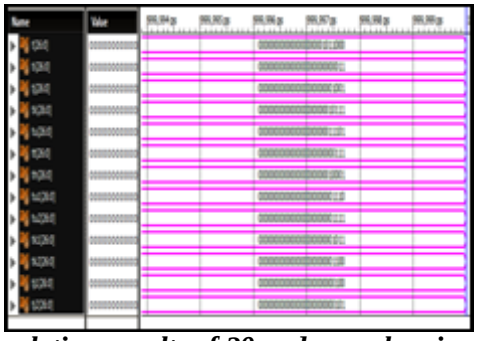


Fig.15 Simulation results of 20 node graph using modelsim

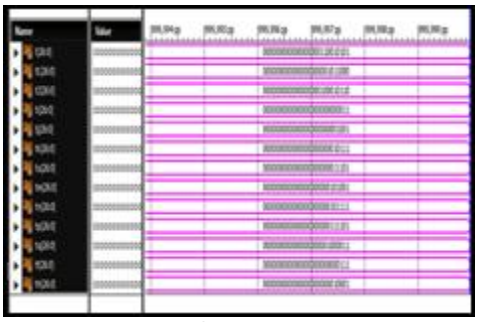


Fig.16 Simulation results of 40 node graph using modelsim

The proposed work of finding the shortest path using prim algorithm has also been synthesized and simulated in cadence tool using Encounter RTL Compiler 9.10 using 180 nm library technology. The dynamic power and leakage power and area occupied for graphs with 3,4,6,10,20 and40 nodes has been acquired and tabulated as shown in below table3.

TABLE.3

Dynamic, leakage and total power and area required for graph with 3,6,10,20,40 nodes.

No of nodes	Leakage Power (µW)	Dynamic Power (µW)	Total Power (µW)	No of Cells	Area of cells
3	0.063	93.969	94.03	41	1677
4	0.241	929.973	930.214	386	8256
6	0.321	935.701	936.022	496	9826
10	0.619	618.174	618.794	806	17936
20	2.325	3561.876	3564.201	2067	64235
40	5.421	8556.810	8562.232	8005	159398

The simulation of the graphs can also be done in cadence tool.The simulation results of the graphs with nodes 20,40 using krushkal algorithm in cadence tool are as shown below.

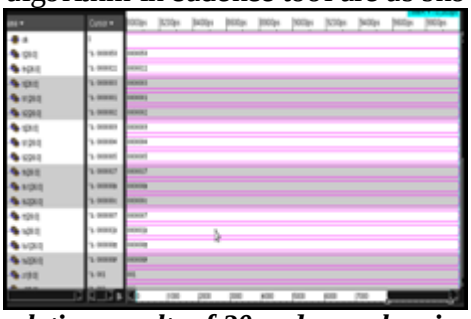


Fig.17 Simulation results of 20 node graph using modelsim

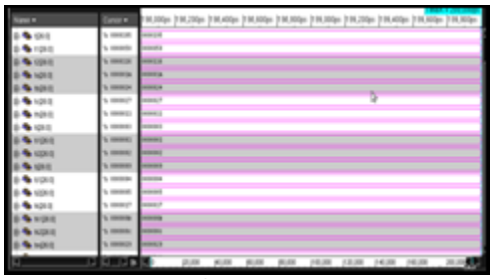


Fig.18 Simulation results of 40 node graph using cadence tool.

B.LAYOUT

The layout of the shortest path computation of graph with 40 nodes using Prim algorithm can be designed in cadence tool and is as shown in below figure.

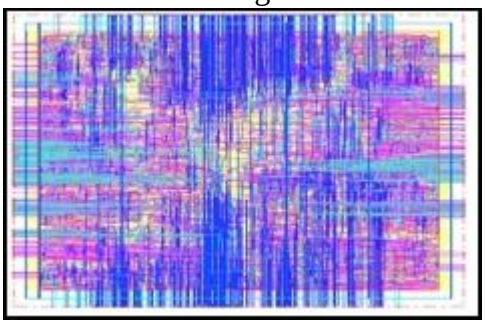


Fig.19Layout of 40 node graph in cadence tool.

IV. HARDWARE IMPLEMENTATION

The proposed work after modelling in verilog code was downloaded in the hardware Spartan -3E and verified with the inputs and its corresponding outputs. The hardware kit comprises 16 in built inputs as the toggle switch and 8 LEDs as output .Using plan ahead 12.2 wizard pin configuration for

input and output was set for the hardware. By the application of boundary scan simulated model of the design was written and finally implemented.The glowing of the leds indicate the selection of the paths 01,10 among the three paths 01,10,11 of a three node graph.These selected paths have to be added to form a shortest path of the graph.

Fig.20 Hardware implementation of 3 node graph using Spartan -3E

VI.CONCLUSION

The shortest path among the node of networks like mobile,road, wireless sensor, etc,can be achieved in an efficient way by utilizing the presented design.Even its utility can be expanded to the larger circuitwhich can be relegated to 40 nodes to find the shortest path.The shortest path for much larger network can be achieved by scaling it down to 40. In the same way, the application were node requirements are of scarce amount minimal node shortest path model can be put to use which dissipates less power. The mesh of lesser node can be further connected in tandem with another mesh. Thereby, building an efficient system of shortest path computation.The prim and krushkal algorithm are written in verilog code and simulated in Xilinx software. The resources required by these algorithms for various graphs consisting of different nodes is acquired by Xilinx and cadence tool .The layout of finding the shortest path using prim algorithm for a graph of 40 nodes is designed. The shortest path for 3 node graphis found in Spartan-3E kit, where the blinking leds indicate the shortest path. The number of LUTS required for a 40 node graph is 310 where as existing design required 2454 number of LUTS. Therefore the resources required is optimised in the proposed model.

REFERENCES

1. G.R. Jagadeesh T. Srikanthan C.M. Lim -*Field programmable gate array-based acceleration of shortest-path computation*
2. Nadira Jasika, Naida Alispahic, Arslanagic Elma, Kurtovic Ilvana, Lagumdzija Elma, Novica Nosovic- *Dijkstra's shortest path algorithm serial and parallel execution performance analysis*
3. R.Nallusamy a K.Duraiswamy b D.Ayya Muthukumar c and C.Sathiyakumar – *Energy efficient dynamic shortest path routing in Wireless Ad hoc Sensor Networks using Genetic Algorithm*
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: *'Introduction to algorithms'* (The Massachusetts Institute of Technology, 1990)
5. Ahuja, R.K., Mehlhorn, K., Orlin, J.B., Tarjan, R.E.*: 'Faster algorithms for the shortest path problem'*, J. ACM, 1990, 37, (2), pp.213–223
6. Nilsson, N.J.: 'Problem solving methods in artificial intelligence' (McGraw Hill, 1971)
7. Cherkassky, B.,Goldberg,A., Radzik,T.: *'Shortest paths algorithms: theory and experimental evaluation'*, Math. Program., 1996, 73, pp. 129–174
8. *Crauser, A., Mehlhorn, K., Meyer, U., Sanders, P.: 'A parallelization of Dijkstra's shortest path algorithm'*. Proc. 23rd Symp. on Mathematical Foundations of Computer Science, 1998, vol. 1450, pp. 722–731
9. Meyer, U.: *'Buckets strike back: improved parallel shortest-paths'*. Proc. 16th Int. Parallel and Distributed Processing Symp., 2002
10. Driscoll, J.R., Gabow, H.N., Shrairman, R., Tarjan, R.E.: *'Relaxed heaps: an alternative to fibonacci heaps with applications to parallel computation'*, Commun. ACM, 1998, 31, (11), pp. 1343–1354
11. Korf, R.E.: *'Planning as search: a quantitative approach'*, Artif. Intell., 1987, 33, pp. 65–88
12. Car, A., Frank, A.U.*: 'General principles of hierarchical spatial reasoning – the case of wayfinding'*. Proc. Sixth Int. Symp. on Spatial Data Handling, 1994, vol. 2, pp. 646–664
13. Jagadeesh, G.R., Srikanthan, T., Quek, K.H.: *'Heuristic techniques for accelerating hierarchical routing on road networks'*, IEEE Trans. Intell. Transp. Syst., 2002, 3, (4), pp. 301–309

