International Refereed Journal of Engineering and Science (IRJES)
ISSN (Online) 2319-183X, (Print) 2319-1821
Volume 5, Issue 2 (February 2016), PP.12-16

Sliced BFS

P. Michael Preetam Raj
(Asst. Professor, ECE dept, K L University, India)

P. Ashok kumar[1], P.G.R.Alekhya[2], K.L.Manasa[3], G.S.Spandana[4]
(ECE, K L University, India)

Abstract:- For Very Large Scale Integration (VLSI) schematics and layouts the Structural verification is formalized. Both schematics and layouts are modeled as graphs and their structural accuracy is tied to a precise set of graph composition rules for defining how these blocks of schematics and layouts may be composed. Verification techniques like Novel, non-heuristic are introduced which allow structural verification for a series of schematic and layout block sizes to be performed. By means of one effective structural verification mechanism, an amalgamated approach to schematic design style verification, layout design rule verification and schematic vs. layout comparison can be provided. These verification techniques are nimble and can be performed with accession as the schematics and layouts are formed. In this we are implementing SBFS by slicing the queue at each level in the tree. The searching technique is implemented in parallel in both the queues. By this the number of searches is reduced and the time taken is half that is required for BFS as drawing the path is not done. This technique works efficiently if the goal node is after the middle node in each level.

Keywords:- Breadth first search, design rules, graph, schematics, structural verification, Top down approach.

I. INTRODUCTION

BFS is one of the techniques for traversing a tree or graph. It starts from a source or root node and explores all neighboring node before going to next level. It performs operations using a queue instead of a stack. The delay for dequeue is reduced by checking a vertex is discovered or not before enqueue operation. Then the time and space complexity of the algorithm is O(bd) where d is the depth of the solution and b is the branching factor (i.e., number of children) at each node.

Earlier graph searching techniques took more time as they were based on series searching i.e. node after node SBFS reduces this time consumption. Here we explain the technique behind SBFS and algorithm to search a node reducing timing constraints.

II. GRAPH TERMINOLOGY

A graph is a pair of sets G = (V, E), where V is a set of vertices, and E is a set of pairs of distinct vertices called edges. A vertex u is adjacent to a vertex v if (u, v) is an edge, i.e., (u ,v) ∈ E. The set of vertices adjacent to v is Adj(v). An edge E = (u, v) is incident on the vertices u and v, which are the ends of e. We use K_n to denote such a graph. A graph H is called the complement of graph G = (V, E) if H = (V, F), where, F=E ($K_{|V|}$) – E. The node from which the search starts is known as root node and the node in the search tree having no children is considered as leaf node.

III. IMPLEMENTATION

SBFS implementation is based on Top down approach. A top down approach designs a strategy for the program. We dwell on the major steps or subtasks in a program. As we enhance each of these steps, we come close to a point at which the algorithm for allthese steps becomes both corporeal and easily adaptable to a particular programming language.

Once we determine the major subtasks for the program as a whole, we in effectcan divide the problem into several smaller programs. As these subtasks are so simple it is possible that we can straight away see how to write the code to accomplish thesetasks. This type of design methodology results in a "tree-like" structure. While the trunk of a tree divides into a number of large branches, every large branch splits into smaller branches and each small branch splits into twigs, the modules at each level spawns several modules at the next level in a top down design and each one of these gives rise to further modules at the next level.

In SBFS, we are considering all the nodes(n) in an each level and taking them in two queues for searching parallel .Two queues are q1 and q2, where q1 has a length of $\frac{n+1}{2}$ if n is odd and $\frac{n}{2}$ if n is even and q2 has a length of n-(q1_length).Both the queues are searched parallel at the same time. so the searching time is reduced than the other techniques. The time taken to search for a goal node is almost half of the time taken for BFS. Write the algorithm for each step (function) in order.

IV. ALGORITHM

```
        sbfs (n,l,q1_length ,q2_length)
        {
                 N: no.of levels in a tree;
                For (L=0;L<=N-1;L++)
                {
                        n=total no.of nodes in L level;
                        if(n is even)
                        q1 length=(n/2);
                        else
                        q1 length=((n+1)/2);
                        q2 length=n-(q1 length);
intQ1[q1_ length];*//* define queues Q1,Q2*//*
intQ2[q2_ length];
Place the nodes in Q1 and remaining nodes in Q2;
While(Q1,Q2 are not empty)
For(x=0;x<=q1_ length;x++)
{
If(Q1[x]==goal node)
                goal node found;
                stop
Else if(Q2[x]==goal node)
Goal node is found;
Stop
        Else
Goal node not found;
}
      }
}
```

Let us explain the algorithm in detail with the help of an example. consider a graph of 5 levels as shown in Fig 1, each node consists of two child nodes at each level.As we discussed earlier two queues are considered namely q1 and q2.length of q1 and q2 varies at each and every level as level value is being updated for every iteration.

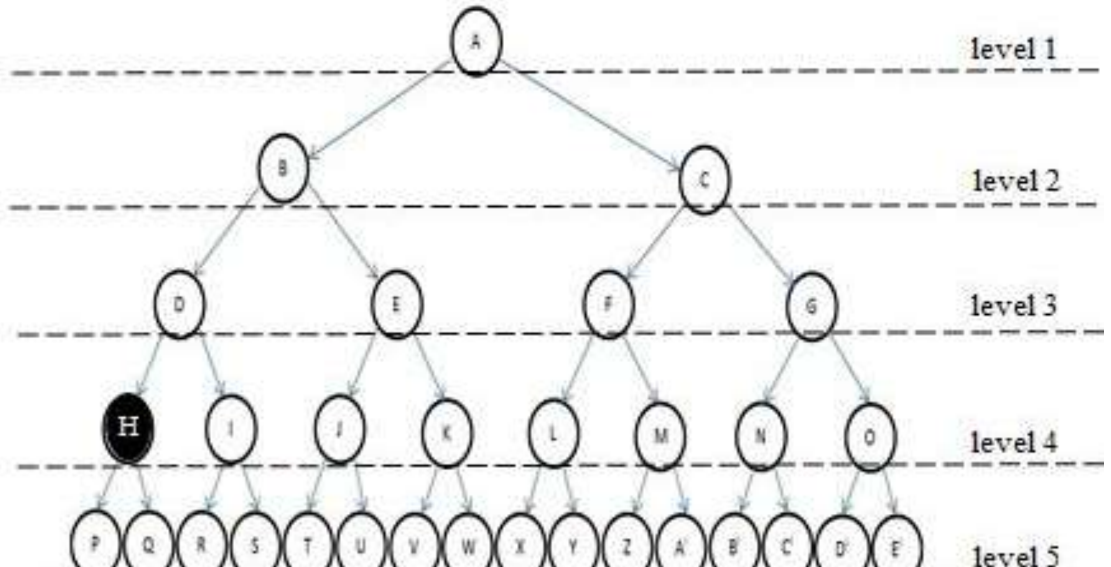

Fig 1: Representation of a graph

In step1,level 1 is considered which is having only one node A,therefore q1 length is one.Let us consider that H is our goal node. A is compared with H , search is performed in one unit time within one iteration and returns failure. In step 2, level 2 is considered with two child nodes of A namely B and C,length of q1 is 1 and q2 is also one. Searching is done in 1unit of time within one iteration and returns failure, for two levels searching has been performed with in two iterations. It continues until goal node is found. All the steps are illustrated in the below Fig 2.

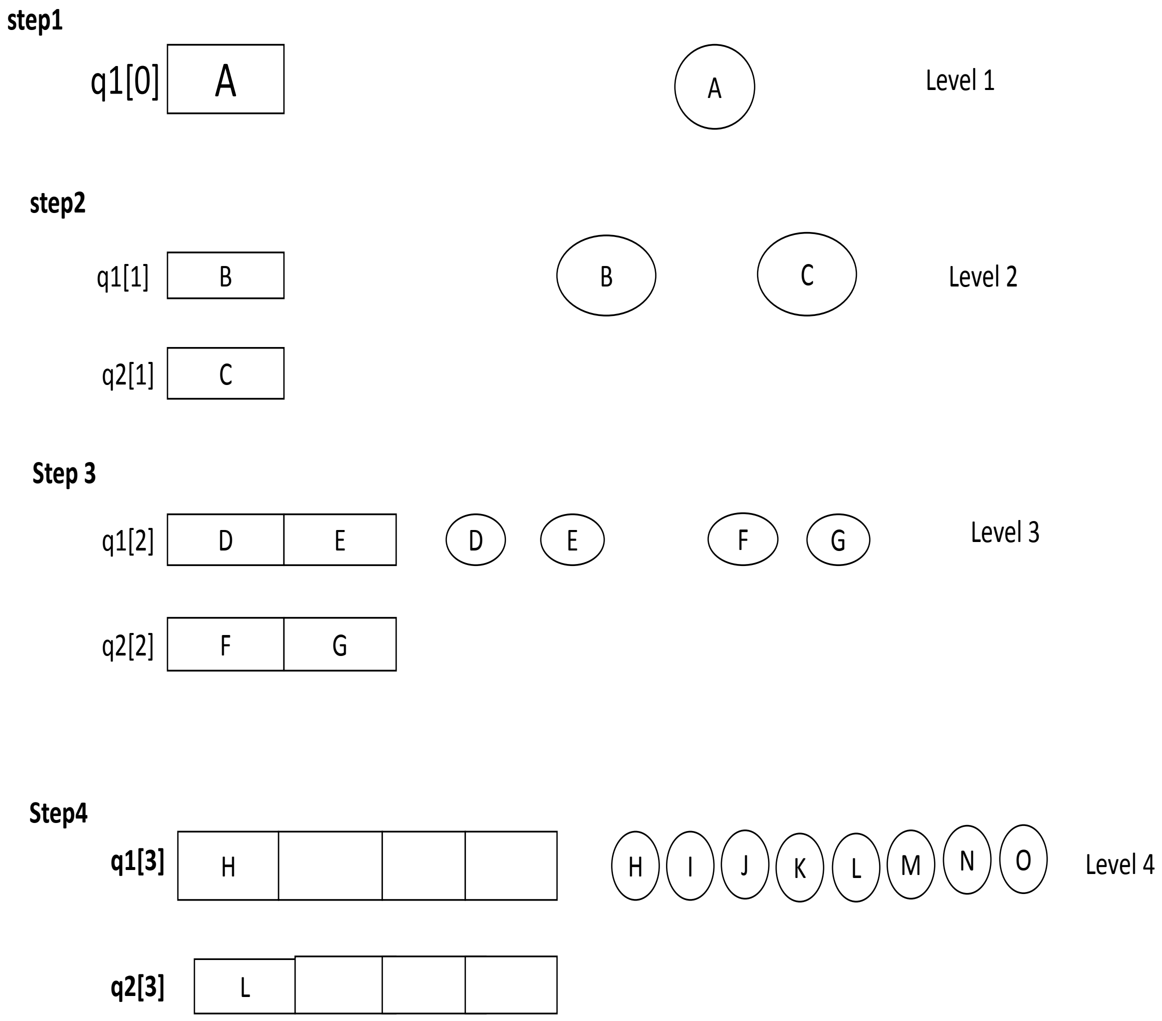

Fig 2: Step wise algorithm explanation

V. TIMING CONSTRAINTS

At each clock pulse simultaneously both queues q1 and q2 are searched.Output will be high if search is success i.e. if goal node is found in any one of the queues; whereas the output will be low if search is a failure i.e. goal node is not found in any one of the queues.

Let us explain it with an example, consider the graph in above Fig 2. At first clock pulse q1 has A and Q2 has D which aren't our goal nodes so, corresponding output will be low. Similar searches are performed until our goal node is found.

At an instant of time i.e. at seventh clock pulse q1 is having H and q2 is having M where His our required goal node therefore, the corresponding output at the seventh clock pulse is high.

The timing diagrams are shown in below Fig 3.

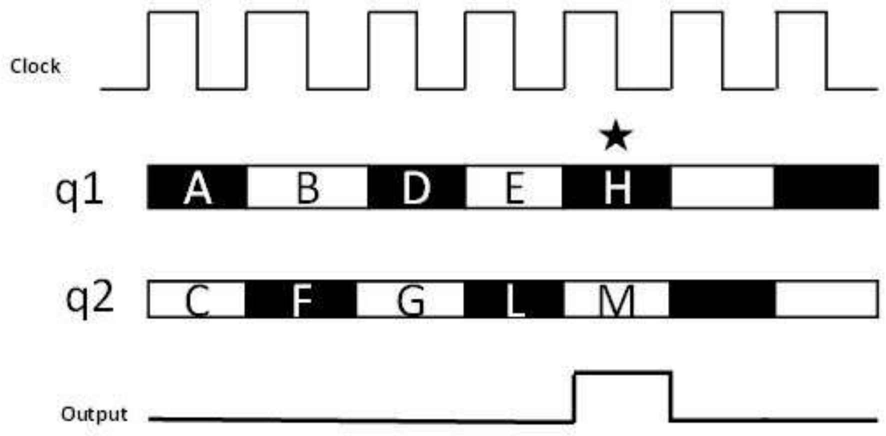

Fig 3: Timing Diagram

VI. CONCLUSION

In this paper, we discussed about a new searching technique using breadth first search by slicing the graphs in much less time compared to the various other techniques being used. In the previous illustrated algorithms so far, searching is done by means of serial search node by node in each and every level, which takes much time.it was overcome by SBFS in which searching is done parallel and which searches nodes in less amount of time. The technique is explained in detail.Also, we developed an algorithm and derived the result using timing constraints.The results of this work demonstrate the performance improvement potential of integrating the top down approach into BFS. This work presents an algorithmic innovation to accelerate the processing of more difficult-to-parallelize BFS.

REFERENCES

[1]. Naveed A. Sherwani, Algorithms for vlsi physical design automation, Intel corporation, (New York, Kluwer Academic Publishers, 2002) 97-149.

[2]. Sabih H. Gerez, Algorithms for Vlsi Design Automation, Wiley, (England, John Wiley & sons, 1999), 21-37.

[3]. Alan Mislove et al. Measurement and analysis of online social networks. ACM SIGCOMM Conference on Internet Measurement (IMC), 2007.

[4]. David Mizell and KristynMaschhoff. Early experiences with large-scale Cray XMT systems. International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2009.

[5]. Duane Merrill, Michael Garland, and Andrew Grimshaw. Scalable GPU graph traversal. Principles and Practice of Parallel Programming, 2012.

[6]. Duncan Watts and Steven Strogatz. Collective dynamics of 'small-world' networks. Nature, 393:440– 442, June 1998.

[7]. GrzegorzMalewicz et al. Pregel: A system for largescale graph processing. International Conference on Management of Data (SIGMOD), Jun 2010.

[8]. HaewoonKwak et al. What is Twitter, a social network or a news media? InternationalWorld Wide Web Conference (WWW), 2010.

[9]. JurijLeskovec et al. Realistic, Mathematically tractable graph generation and evolution, using Kroneckermultiplication. European Conference on Principles and Practice of Knowledge Discoveryin Databases, 2005.

[10]. Paolo Boldi others. Ubicrawler: A scalable fully distributed web crawler. Software: Practice &Experience, 34(8):711–726, 2004.

[11]. Wikipedia page-to-page link database 2009. http://haselgrove.id.au/wikipedia.htm.Christo Wilson et al. User interactions in social networks and their implications. European conference onComputer systems (EuroSys), 2009.

[12]. Y Low et al. GraphLab: A new framework for parallel machine learning. Uncertainty in Artificial Intelligence, 2010.

[13]. Convey HC-1 family. www.conveycomputer.com/ Resources/Convey_HC1_Family.pdf.

[14]. Timothy Davis and Yifan Hu. The University of Florida sparse matrix collection. ACMTransactions on Mathematical Software (to appear), cise.ufl.edu/research/sparse/matrices.

[15]. Sungpack Hong, TayoOguntebi, and KunleOlukotun. Efficient parallel graph exploration on multi-core CPU and GPU. Parallel Architectures and Compilation Techniques (PACT), 2011.

[16]. HaewoonKwak et al. What is Twitter, a social network or a news media? International World Wide Web Conference (WWW), 2010.

[17]. JurijLeskovec et al. Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. European Conference on Principles and Practice of Knowledge Discovery in Databases, 2005.
[18]. Y Low et al. GraphLab: A new framework for parallel machine learning. Uncertainty in Artificial Intelligence, 2010.
[19]. GrzegorzMalewicz et al. Pregel: A system for largescale graph processing. International Conference on Management of Data (SIGMOD), Jun 2010.
[20]. Duane Merrill, Michael Garland, and Andrew Grimshaw. Scalable GPU graph traversal. Principles and Practice of Parallel Programming, 2012.
[21]. Alan Mislove et al. Measurement and analysis of online social networks. ACM SIGCOMM Conference on Internet Measurement (IMC), 2007.
[22]. David Mizell and KristynMaschhoff. Early experiences with large-scale Cray XMT systems. International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2009.
[23]. Paolo Boldi others. Ubicrawler: A scalable fully distributed web crawler. Software: Practice & Experience, 34(8):711–726, 2004.
[24]. Duncan Watts and Steven Strogatz. Collective dynamics of 'small-world' networks. Nature, 393:440–442, June 1998.
[25]. Wikipedia page-to-page link database 2009. http://haselgrove.id.au/wikipedia.htm.
[26]. Christo Wilson et al. User interactions in social networks and their implications. European conference on Computer systems (EuroSys), 2009.
[27]. Andy Yoo et al. A scalable distributed parallel breadthfirst search algorithm on BlueGene/L. International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2005.
[28]. Kisun You et al. Scalable HMM-based inference engine in large vocabulary continuous speech recognition. IEEE Signal Processing Magazine, 2010.