Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Study of Data Flow Graph Representation Analysis with Syntax and Semantics

Pushpendra Singh[1], Devesh Chaurasiya[2], Ankita Joshi[3], Kumar Sambhav Pandey[4]
Department of Computer Science and Engineering
National Institute of Technology
Hamirpur (H.P.)

Abstract- ***Flow of data in graphical representation form gives a modern style in presenting the program statements. The syntax and semantics of data flow graph provides the opportunity to explore their efficiency in field of synthesis and verification of architecture system. The execution of operations on the nodes takes place through firing rule makes data flow graph reliable in terms of some parallel execution. This paper presents the data flow graph format on the basis of different nodes used in various loop construct conditions of program. The semantical analysis of nodes are explained with the help of examples. In the last, this paper also propose an approach for using dummy node in data flow graph.***
Keywords- data flow graph, firing rule, semantics, token.

I. INTRODUCTION

Data flow graph concept gives an idea in presenting the problems or programs in a graphical representation form. The programmer wants to show his capability by understanding all the related information of problems and put his ideas in an ideal form but sometimes it can't express and achieve expected performance in his results. So data flow graphs provide a representation platform with simple and attractive interface.

In this paper, semantics and specifications of data flow graph is presented. Data flow graphs are well known and good modelling tool for the behavioural specification of a system, as in ([1], [2]). Data flow graph format having both syntax and semantics simple interface. Analysis of data flow graph helps in exploring global information about the manipulation of data for a procedure or a larger program, as in [3]. It shows the working, how it manipulates its data in their execution process.

Data flow programs are represented in the form of graph. For solving the problems of larger programs data flow graphs are easily arrangeable. By eliminating the need to manage parallel execution explicitly, data flow graph representation exposes the maximum parallelism for given program, as in [4]. Data flow graph only depends on data availability at input and not preferring the specific order of execution. Token passing method based semantic used for describing the flow of data between nodes in data flow graph through their edges between them. *Token* is defined as a single data value instance at the input of a particular node at the time of execution of operation, as in [5].

II. DATA FLOW GRAPH

The graphical representation form which consists of nodes and directed arcs called *data flow graph*. In mathematical terms we say that, a graph G (V, A), where V= (v_1,v_2……,v_n) is the set of nodes and A=(a_1,a_2,……..,a_m) Є VXV the set of directed arcs or edges a_i=(v_j,v_k), as in [6]. The nodes represents instructions and arcs represents the data which will exchange between these nodes.

The nodes of data flow graph perform operations when the data value or token available at its input port. The execution of operation according to the firing rule. *Firing rule* says that node only executed when the tokens are available at all input ports, as in [7]. The process of execution proceed with the fetching of tokens form the input port and result will be forwarded through each output port. Every node connected through one of the node in the network of graph, so producing node send the data to the consuming node through edges. Data flow graph follows the partial ordering execution of nodes. Due to the partial ordering in the data flow graph nodes are not strictly dependent on each other for their execution of instruction.

III. SYNTAX AND SEMANTICS

Data flow graph uses different types of nodes for the synthesis and verification of the programs. The semantics of data flow graph helps in providing unique degree of freedom. The programmer uses the simple interface of data flow graph to produce optimized solution for the given system. The syntax and semantics are described for different nodes of data flow graph so that nodes behaves efficiently in various conditions of program statements. Some of the basic nodes are described here through which we will construct solution without

following any restrict order of execution. The type of nodes are:

A. Operator nodes

A large number of operation performed on the nodes. Some of the basic arithmetic operations are +, -, x, *, or boolean like >, <, =, or , and, or some of the complex functions, as in [6]. The graph represents in fig.1 shows the following statements :

```
int b, x, y, z ;
main ( )
{ b= 3*y* y  + y  + 1; z= x+ b; }
```

The data flow graph format allow the nesting of graphs with absorption of results of one node to another node as input. The operator used in the data flow graph for performing various operation are not fixed only depends on the input ports. The input ports defines that how many number of inputs are using for performing particular execution.

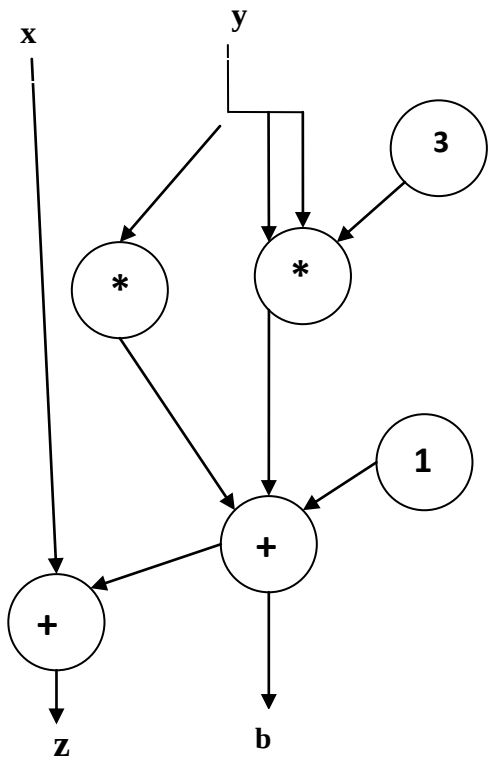

Fig. 1. Data flow graph for $b= 3y^2+y+1, z= x+b$

B. Input –Output nodes

Input and output are the basic node to perform some logical operation on any node. Input node defined as a nodes of type input are the only nodes without input ports. Output node defined as a nodes of type output are the only nodes without output ports, as in [5]. When the token is available on each input node then execution of process started and give accordingly result through its output node.

C. Get- Put nodes

For the complex program representation input and output nodes are enhanced as get and put node to read and write the operation on the port. When the interaction through outside world take place then get and put nodes easily occur inside the loop. Get and put node easily manage all the inputs and outputs which are going into charge during the complex program execution.

D. Branch nodes

Branch node is mainly used for the representation of conditions occur in the program. When one condition terminates and control jumps to another condition then branch node comes in focus in the data flow graph. At particular branch node, decision will be taken on the basis of control token value as true or false. The branch node fires its execution when all its input are available on input port and result will be forwarded to the output port using the control token value.

E. Merge nodes

Merge node act as complementary to the branch node in the data flow graph. Merge node has one important characteristic which make different from the branch node. Merge node not follow the firing rule concept in a strict manner. It execute its operation without availability of all tokens at its input port, only based on control token value.

Sometimes merge node only passes the value received from its predecessor to one of its output port. In this case merge node fires without using the control value only the token availability on the input port is the main consideration.

The main utilization of branch and merge node for solving the problems which include like if-then-else or case like conditions. For clarification of this concept take an example:

```
If (m) then
   {
      y = y ;
      x = x + 2 ;
   }
else
      y = y ++
      x = x - -
```

The data flow graph of above program is shown in fig. 2

© 2012, IJARCSSE All Rights Reserved

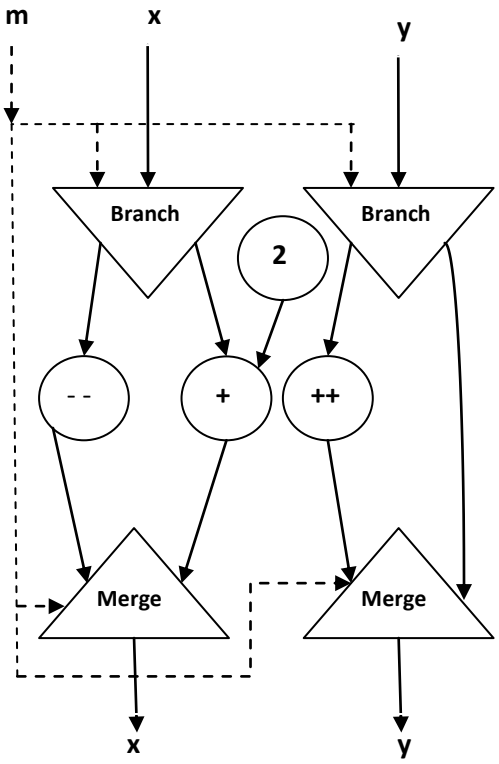

Fig. 2. Data flow graph for if- then- else.

F. *Exit- entry node*

Exit and entry nodes takes the logical idea from branch and merge node respectively. The working of exit and entry node in a given program representation uses the similar functionality as branch and merge node. The token is available at control input for every entry nodes for the initialization of execution of process and result is passing through the exit node. For different values of token at input of entry node the process is executed repeatedly.

The main focus of entry and exit node to interact with loop construct problems. The loop programs include like while…do or for…do are represented by the exit and entry node in data flow graph. The following example shows the loop constructs:

```
add (int x, int y)
  {
    while ( x >= y )
        x= x + y ;
    return (x) ;
   }
```

The fig.3 shows the graphical representation of this program:

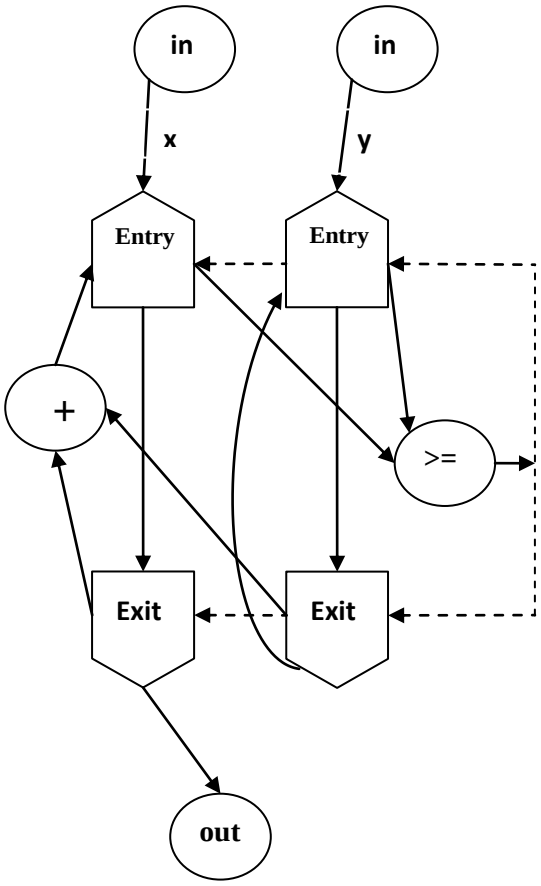

Fig. 3. Data flow graph for while loop

G. *Constant nodes*

For providing constant values constant nodes also counted in these nodes category. In program statements sometimes only fixed constant data value is passing in various loop constructs. Constant data value is added at particular condition for this we specify constant node at that position in the data flow graph. The main feature about constant node that it has only one output port. Constant node uses this output port only for passing the fixed data value on the basis of control value.

Here we can analyse that all the above nodes are basic nodes, used for the representation of data flow graph . All the nodes have some semantics which show their efficiency or usefulness under well defined conditions. Basically all nodes of data flow graph follow the firing rule for starting their proper execution. The important feature about data flow graph is that, at particular time more than one node fires on the basis of the availability of tokens at input port.

IV. ANALYZING DATA FLOW GRAPH WITH CONTROL FLOW GRAPH

A possible flow of continuous program statements as basic blocks are represented with nodes and edges by control flow graph. Each node is considered as a basic blocks in control flow graph. Entry block is used for entering of controls and exit block for leaving the control as result. Control flow graph dependent on the control flow between the nodes hence it is

© 2012, IJARCSSE All Rights Reserved

control dependent, as in [8]. In control flow graph sequential ordering occur for the execution of process. Control flow graph mainly consists of two types of references. These two references are control flow and data flow. Due to the control dependency, control flow graph does not explore efficiently for hardware design implementations.

Data flow graph dependent on the flow of data between the nodes. In other terms data flow graph is data dependent, as in [8]. The execution of process follows the partial ordering between the nodes. Due to partial ordering a large number of nodes executed at particular time. The execution node only depends on the resources or data value availability at that time for the nodes which are ready for execution. The representation of data flow graph uses some semantics as different types of nodes in different loop construct conditions. In this way data flow graph provides better platform for hardware design implementations.

V. IMPLEMENTATION APPROACH

Dataflow graph uses the different nodes for their representation by following some syntax and semantics. The generated graph for given program with the help of these nodes provides the opportunity to explore our ideas in a new direction. Data flow graph does not include the control arcs because it is data dependent. By using this concept from control flow graph we generate data flow graph by exploring each basic block of control flow graph. The restriction of using program counter in control flow graph easily eliminated into the generated data flow graph. The standardization of generated data flow graph can be done with the help of using dummy nodes. Dummy node is a type of additional node for making equal number inputs and outputs at particular node. During the splitting of nodes at particular level, the dummy node plays important role. The implementation of data flow graph with dummy nodes helps in exploring control flow graph for architecture synthesis of hardware. In this way execution takes place in a similar standard manner for each node in the graph. So, we easily trace out the whole graph easily and also implement for some hardware designs.

VI. CONCLUSION

Data flow graph provides efficient way to present the given program in simple graphical manner. The user easily understand and interact for every possible input and output by using token based mechanism. Firing rule strategy in data flow graph increase the certainty of parallel process execution. In this way data flow graph are highly suitable for synthesis and verification of the architectural based system. The synthesis done with help of graphical view of data flow graph provides many alternative ideas and implementation designs for particular program in a very simple manner. The partial order of execution of data flow graph also helps to increase its flexibility and parallelism factor for particular program.

ACKNOWLEDGMENT

This work was supported in part by Ministry of Human Resource Development (MHRD) and the Department of Computer Science and Engineering, N.I.T. Hamirpur (H.P.).

REFERENCES

[1] Davis, A.L. and R.M. Keller, "Data Flow Program Graphs", IEEE Computer, vol.15, no. 2, pp. 26-41, February 1982.

[2] K.M Kavi, B.P. Buckles, and U.N. Bhat, " A Formal Definition of Data Flow Graph Models", IEEE Trans. Computer, vol. c-35, no. 11, pp. 940-948, November 1986.

[3] Steven S. Muchnick, Advanced Compiler Design and Implementation, Morgan Kaufmann, 1997.

[4] B Lee, A.R. Hurson, "Dataflow Architectures and Multithreading", IEEE Computer Society , vol. 27,issue 8, August 1994.

[5] Jos T.J. Van Eijndhoven, L. Stok, " A Data Flow Graph Exchange Standard", Design Automation , Proceedings of the European Conference, March 1992 .

[6] Gjalt G. de Jong, "Data Flow Graphs: System specification with most unrestricted semantics", Design Automation. EDAC., Proceedings of the European Conference, February 1991.

[7] Arthur H. Veen, " Dataflow Machine Architecture", ACM Computing Surveys (CSUR), vol. 18, issue 4, December 1986.

[8] Gang Quan, "Data Flow Graph Intro," Maseeh College of Engineering and Computer Science, Portland State University, http://web.cecs.pdx.edu/~mperkows/temp/JULY/data-flowgraph. pdf .

[9] S. Amellal, b. Kaminska, "Scheduling of a Control and Data Flow Graph", Circuits and Systems, IEEE International Symposium, vol. 3, May 1993.

[10] G Xue-rong, Z Rong-cai, L lin-sheng, "Communication Optimization Algorithms based on Extend Data Flow Graph", Eighth ACIS International Conference ,vol.3, August 2007

© 2012, IJARCSSE All Rights Reserved