An Intelligent Priority Decision Making Algorithm for Competitive Operators in List-based Scheduling

Jun-yong Lee† and Sunil Kim†

†Department of Computer Engineering, School of Engineering, Hongik University, Seoul, Korea 121-791

Summary
In this paper, a new intelligent algorithm was proposed to minimize the number of control steps of overall operation when scheduling is performed by deciding the precedence of competing operations under the condition that only limited hardware resources were allowed to use. The proposed method could reduce the number of control steps for operation by using multiple criteria, while previous algorithms use one simple criterion. To combine several criteria, 'Fuzzy Logic' was used which allows continuous logic values between 0 and 1 unlike the traditional Boolean logic. To show the performance of proposed algorithm, a number of randomly generated test cases were used for simulation.

Key words:
Scheduling, Computer Architecture, Fuzzy Logic, High-level Synthesis

1. Introduction

As the structures of computer hardware become more complicated and the degree of circuit integration becomes higher, the design of hardware system is getting more difficult nowadays. Accordingly, the necessity of software for hardware design has been increased [1], [2].

The computer hardware design consists of many steps. Among them, high-level synthesis is the process which transforms the behavioral description of hardware into a structural description. In this process, inputs are hardware specifications described in HDLs (Hardware Description Languages) such as VHDL or Verilog. Outputs are hardware structures which consist of data paths and the controller which controls the behavior of data paths. High-level synthesis is divided into several steps such as HDL compilation, scheduling, unit selection of components, unit binding of components and so on. Scheduling is an important step because it decides the sequence of operations, which affects the performance of high-level synthesis crucially [10].

In general, scheduling is categorized into two groups. One is the time-constrained scheduling, which tries to minimize the usage of hardware components given a time constraints. The other is resource-constrained scheduling, which tries to minimize the number of control steps using limited hardware components. List-based scheduling is the most popular method of resource-constrained scheduling, where scheduling is performed using a 'priority list' of operations.

Priority list is decided by a priority function, which affects the result of scheduling directly. Therefore, the choice of proper priority function is very important issue for the performance of scheduling. But, it is very difficult to get the optimal priority function since the characteristics of input graphs for high-level synthesis are very complicated and diverse. In our study, fuzzy logic was exploited to combine various priority functions instead of using a single priority function. An efficient method for better result of scheduling was proposed and the result was compared with traditional scheduling method through experiments [6].

2. Related Works

One of the simplest method of scheduling is ASAP (As Soon As Possible) scheduling which places operations in the earliest positions in control step. In contrary, ALAP (As Late As Possible) scheduling places operations in the most backward positions in control step. List-based scheduling is a generalized form of ASAP scheduling with resource restriction [7].

2.1 List-based Scheduling Algorithm

In the list-based scheduling algorithms, priority lists are used, which include operations to be performed. In this list, operations are stored in the order of priority, where the operations are ready to be scheduled, which means preceding operations are completely scheduled already. Once the priority list is ready, the scheduler places operations in the control steps in the order of priority. This

Manuscript received January 5, 2009
Manuscript revised January 20, 2009

allocation continues until the given hardware resources are exhausted.

Various methods have been studied about the priority functions. Some of them are mobility range of operations, the length of critical path, repelling force between operations and so on [3].

2.2 Fuzzy Logic

Fuzzy logic introduced by L. A. Zadeh [5] in 1965 has been used in various applications of intelligent heuristics implementation.

A classical (crisp) set is normally defined as a collection of elements or objects. For a given set $A \subseteq X$, each element $x \in X$ either belongs to the set A or does not belong to the set A, where X is a set of elements denoted generically by x. But fuzzy sets are different from classical sets in that they allow partial or gradual degrees of membership. In other words, classical logic allows either 'True' (1) or 'False' (0) as a logic value. But fuzzy logic allows any value between 1 and 0 as a logic value.

For example, a statement "Tom is young." can be either 'True' or 'False' (1 or 0) in traditional logic. But in fuzzy logic, the statement "Tom is young." can have a value between 0 and 1. This value is decided by a membership function. Figure 1 provides an example of the membership function 'young' defined graphically for the linguistic variable 'age.'

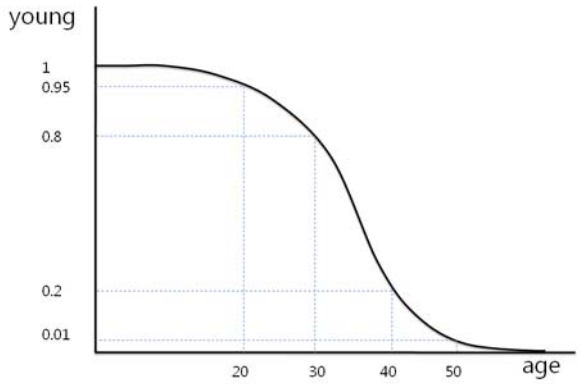

Fig 1. Membership function 'young' for 'age.'

A person whose age is 40 has a degree of membership 0.2 in the fuzzy set 'young person.' In other words, if Tom is 40 years old, a statement "Tom is young" has a logic value of 0.2. If Peter is 20 years old, a statement "Peter is young." has a logic value of 0.95. Like this, logic value can be any number between 0 and 1, as it is called 'fuzzy.'

2.3 Decision Making Using Fuzzy Rules

Fuzzy decisions are made using plain language rules describing relations between linguistic variables and their values defined by membership functions. Rules are stated in a simple IF/THEN format, and input values can be combined using **and** or **or** operations. *Min* and *Max* are most commonly used functions for fuzzy *intersection* (*and*) and *union* (*or*) operators, but other functions were also proposed [5] for more sophisticated applications.

For example, let us assume that a fuzzy rule is stated as follows:

If a person is young and strong,
then the person is a good candidate for an athlete.

Let us assume that fuzzy sets 'young person' and 'strong person' are given as follows for Peter, Mary and Tom,

A = 'young person'
= { (Peter, 0.95), (Mary, 0.5), (Tom, 0.2) }

B = 'strong person'
= { (Peter, 0.3), (Mary, 0.7), (Tom, 0.6) }

Then a fuzzy decision can be made by combining the two fuzzy sets. If min is used for fuzzy intersection, then we get

A ∩ B = 'young and strong person'
= { (Peter, 0.3), (Mary, 0.5), (Tom, 0.2) }

Therefore, Mary is chosen as the best candidate for an athlete among them.

3. Priority Decision Making for Competing Operators

3.1 Drawbacks of Previous Scheduling Methods

In the previous scheduling methods, a single priority criterion was used to schedule the operations. In this manner, if the priority is same for more than two operations, the priority is decided randomly for these competing operations.

Following example is a result of list-based scheduling where 3 multipliers, 1 adder and 1 subtracter are available hardware resources. Mobility range of operation was used as criterion for priority decision.

Figure 2 represents an input DFG (Data Flow Graph) to be scheduled. Operations V0 and V1 are ready to be scheduled as there is no prior operations. Since only one adder is available, the two operations V0 and V1 are competing to be scheduled. Mobility range is 0 for the two operations.

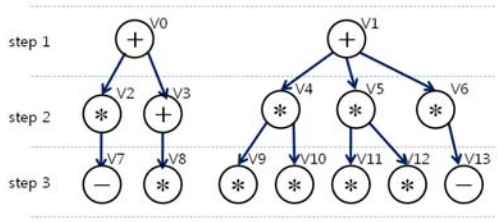

Fig 2. DFG(Data Flow Graph) given to the input of scheduling

Figure 3 represents the case when operation V0 was selected first in scheduling. Figure 4 represents the case when V1 was selected. Consequently, better scheduling result was obtained when operation V1 was chosen, which has larger number of following operations.

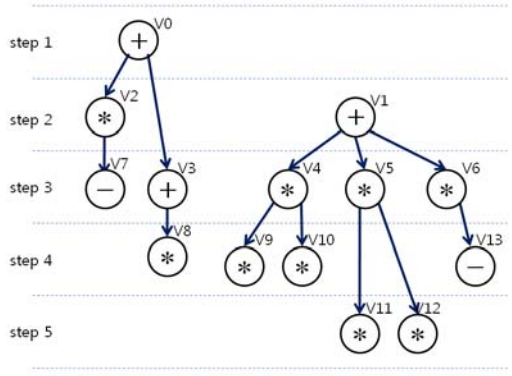

Fig 3. Case when V0 was selected first

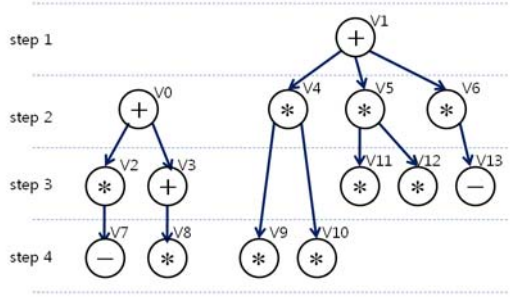

Fig 4. Case when V1 was selected first

3.2 Implementation of Intelligent Algorithm Using Fuzzy Logic

Heuristics, which are generally used for intelligent algorithms, are based on the experience and human knowledge acquired by understanding problems. As fuzzy logic is based on natural languages, it provides convenient methodologies to represent such human knowledge.

Proposed heuristic list scheduling algorithm, which exploits fuzzy logic, performs its task under the resource restriction by considering many constraints such as mobility range of operations, length of the critical paths, repelling force of operations, and the number of following operations.

Since the characteristics of input DFGs for list scheduling are various, the optimal priority function does not exist. Therefore we have to apply several available priority functions together for better solution. In this study, we implemented an intelligent algorithm which decides priority of operations in such a way that each priority function is defined by membership function of fuzzy logic and used for fuzzy rules.

3.3 Fuzzy Rule for Priority Decision

In order to decide the priority of operations, commonly used four criteria were used to define a fuzzy rule as follows.

IF
"The mobility of a operation is small" AND
"The length of the critical path is long" AND
"The repelling force between operations is large" AND
"The number of following operations is large"
THEN
"The priority of the operation is high."

3.4 Membership Functions

Four basic priority functions used in the fuzzy rule are immobility, criticality, repelling force, and the number of following operations. They were defined by membership functions as follows in piecewise linear functions.

3.4.1 Immobility

Immobility was defined as the reverse of mobility of operations. If the mobility of an operation is small, the immobility becomes large. The maximum value of mobility of operation in the set of DFG used in experiment is defined by 1. The relative mobility of each operation was computed as the ratio over the maximum value of mobility. The membership function is denoted in Fugure 5.

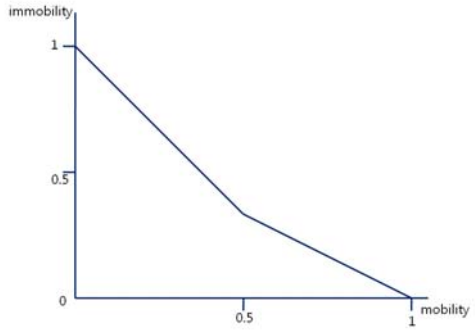

Fig 5. Immobility

3.4.2 Criticality

The criticality of an operation was defined by the length of critical path. The longest critical path was defined by 1, and the relative criticality was computed for each operation.

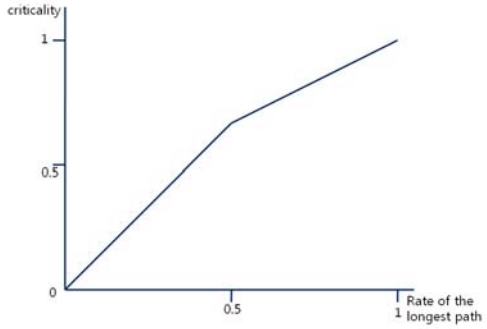

Fig 6. Criticality

3.4.3. Repelling Force

Previously defined repelling force [3] was used in the proposed algorithm as in Figure 7.

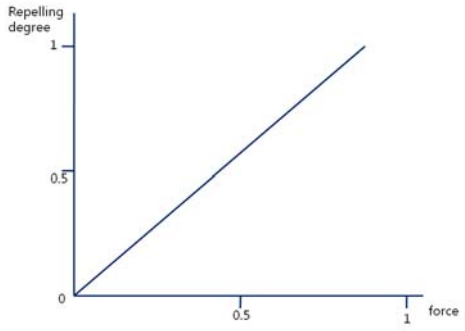

Fig 7. Repelling Force

3.4.4 Amount of following operations

The amount of following operations is defined by the number of following operations from an operation in DFG.

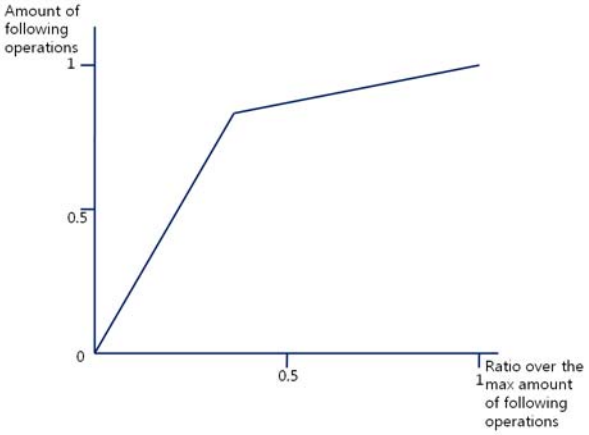

Fig 8. Amount of following operations

4. Proposed Algorithm

Proposed scheduling algorithm is stated as follows in Figure 9. Priority function was defined using fuzzy logic and fuzzy rules.

```
INSERT_READY_OPS(V, PList1, ... PListm)
Cstep=0
while(PList1≠Null)or...or(PList1≠Null) do
   Cstep =Cstep + 1
   for k = 1 to m do
      for funit =1 to Nk do
        if PListk≠Null then
        Select_Op = Compute_Priority_Function(PListk)
        SCHEDULE_OP(Scurrent, Select_OP, Cstep)
        PListk = DELETE(PListk, Select_OP)
      endif
    endfor
   endfor
      INSERT_READY_OPS(V, PList1, ... PListm)
endwhile
```

Fig 9. Proposed algorithm

In the above algorithm, '***Compute_Priority_Function'*** computes the priority of operations using fuzzy logic, where previously defined membership functions are used in fuzzy rules.

5. Scheduling Example

In order to show the basic operation of proposed scheduling algorithm, an input DFG was selected as shown in Figure 10.

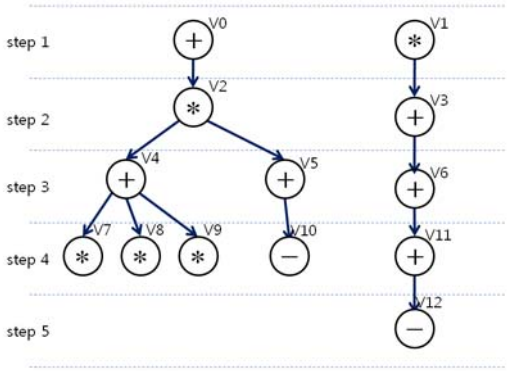

Fig 10. DFG input

Previously defined fuzzy rules were used. It was assumed the available hardware resources are one multiplier, one adder, and one subtracter. The result of scheduling was depicted in Figure 11. It shows that the performance of proposed algorithm outperforms the previous simple scheduling method. In this example, operations V4, V5, V11 compete in step 4. Proposed algorithm selects V4, which decrease the number of overall steps by 1.

	Previous method	Proposed algoritm
step 1	V0, V1	V0, V1
step 2	V2, V3	V2, V3
step 3	V6	V6
step 4	V5	V4
step 5	V10, V11	V8, V11
step 6	V4, V12	V7, V5, V12
step 7	V8, V9	V9, V10
step 8	V7	

Fig 11. Result of scheduling

6. Experimental Results

In our study, schedulers were implemented in C++ to compare the performance of traditional algorithm and proposed algorithm. Input DFGs were randomly generated and the hardware resources were adjusted to compare the results.

Experimental result shows that proposed algorithm can decrease the number of control steps by 10% in maximum. When the available hardware resources are enough, the improvement of proposed algorithm is diminished. In every case, proposed algorithm outperforms traditional method.

Overall result is shown in Figure 12. The resource rate in the graph means the ratio of available hardware over the amount of hardware which does not cause any competition of operations. The values of y-axis denote the ratio of number of steps against the traditional method. The result shows that the number of steps decreses by 10.2% when the resource ratio is 0.2.

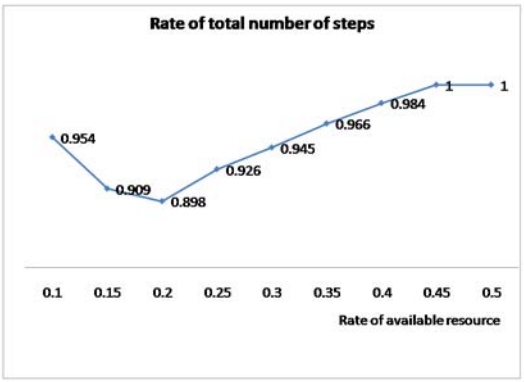

Fig 12. Performance of the proposed algorithm

7. Conclusion

In this study, it has been showed that the scheduling results could be improved by merging several priority functions using fuzzy logic when competing operations are scheduled.

Fuzzy logic provides efficient solutions for intelligent algorithms as it resembles the way of human decision making by allowing continuous logic values between 0 and 1 unlike traditional logic.

Proposed algorithm can be improved more by adopting various fuzzy operations or by using more efficient membership functions. Further study will enable the implementation of better intelligent heuristics. Proposed methodology could be used in various fields other than list scheduling application.

Acknowledgment

This work was supported by 2007 Hongik University Research Fund.

References

[1] Daniel D. Gajski, Nikil D. Dutt, Allen C.-H. Wu and Steve Y. Lin, "High-level Synthesis: Introduction to Chip and System Design," Kluwer Academic Publisher, 1992.
[2] Giovanni DeMicheli, "Synthesis and Optimization of Digital Circuits," McGraw-Hill, 1994.
[3] Pierre G. Paulin and John P. Knight, " Force-directed Scheduling for the Behavioral Synthesis of ASICs," IEEE Transaction on Computer-Aided Design, Vol. 8, pp. 661-679, 1989.
[4] Robert A. Walker and Samit Chaudhuri, "Introduction to the Scheduling Problem," IEEE Transaction on Design and Test, 1995.
[5] H.-J. Zimmermann "Fuzzy Set Theory and Its Application" Kluwer Academic Publishers 1991.
[6] Hojjat Adeli, Kamal C. Sarma. "Cost optimization of structures: fuzzy logic, genetic algorithms, and parallel computing" Willey, 2006.
[7] Oliver Sinnen, "Task scheduling for parallel systems" John Wiley, 2007.
[8] S. Lee, "Advanced digital logic design: using VHDL, state machines, and synthesis for FPGAs" Thompson, 2006.
[9] Wayne Wolf "High-performance embedded computing: architectures, applications, and methodologies" Morgan Kaufmann Publishers, 2007.
[10] A. Peter, "High level synthesis of pipelined data paths" Wiley, 2001.

Jun-yong Lee received the B.E. degree in computer engineering from Seoul National University, Seoul, Korea in 1986. He received the M.E and Ph.D. degrees in computer engineering from the University of Minnesota in 1988 and 1996, respectively. After working as a research staff member in IBM (from 1996), he has been a professor at the Department of Computer Engineering, Hongik University, Seoul, Korea. He also has been a visiting professor at the Towson University in Maryland in 2004. His research interest includes computer architecture, embedded systems, computer security and others. He is a member of KISS, KSCI, IEEK, and KSTC.

Sunil Kim received the B.E. and M.S. degrees in computer engineering from Seoul National University, Seoul, Korea in 1985 and 1987, respectively and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, in 1995. He joined IBM, Austin and worked on PowerPC processor developments from 1995 to 1999. Currently, he is an associate professor at the Department of Computer Engineering, Hongik University, Seoul, Korea. His current research interests include embedded systems, computer security, and system software.