Calculating Error Reduction with Graph Restructuring in Loop Folding

Yoshi NISHITANI Katsumi HARASHIMA Toshirou KUTSUWA
Faculty of Engineering Osaka Institute of Technology
5-16-1 Ohmiya, Asahi-ku, Osaka, 535-8585 Japan
Tel: +81-75-781-3324, Fax: +81-75-781-3324
E-mail: y-nishitani@mwe.biglobe.ne.jp

Abstract: This paper proposes a Data-Flow-Graph (DFG) restructuring to reduce calculating errors in loop folding scheduling. The prime cause of calculating error is rounding errors due to the restriction of the operation digit of functional units. This rounding error is increased more by using multipliers than adders, so reducing the number of multiplications and putting off them as much as possible reduce rounding errors. The proposed approach reduces the number of multiplications by restructuring DFG in loop folding.

1 Introduction

High-Level-synthesis (HLS) is the transformation of a behavioral description into a resister transfer level (RTL) description[1]. HLS decides the execution order of operations (Scheduling), binds the operations to the data-path of hardware and generates a RTL structures (Allocation). The synthesized result influences chip performances and area.

In HLS, a behavioral description is represented such as a Data Flow Graph (DFG), which captures the data dependencies of the given behavioral description.

The purpose of scheduling is to decide the execution sequence of each operation and assign operations to Control Step (CS) under several constraints (e.g. resource costs, power consumption, etc.) for a given DFG. Each CS corresponds to one clock cycle of the CPU executing a given behavioral description.

A behavioral description often includes loop statements. For example, a digital filter repeatedly executes the same set of operations for each sample of the input data stream as an infinite loop[2][3]. In a digital filter, calculating errors influence the characteristic of filtering, so the reduction of calculating errors is essential to design digital filters.

In general, a calculating result corresponds to various behavioral descriptions. This means that a behavioral description can be rewritten to other. The rewriting of a behavioral description is equal to the restructuring of a DFG as shown Fig.1 and Fig.2.

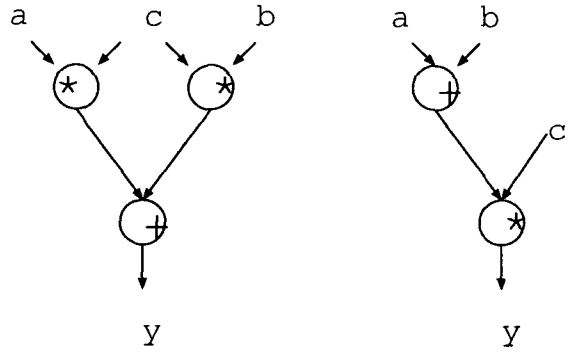
The purpose of conventional approaches in High–Level Synthesis were to minimize the execution time[1], resource cost[4], and power consumption[5], and maximize throughput[6][7]. *Loop shrinking*[6] reduces the interval of output-data to increase throughput by restructuring DFGs including loops. [7] attempts to reduce the critical path length of DFGs by restructuring DFGs.

However, the error in an operation result has not been considered in these previous methods. Therefore this paper has proposed an approach to minimize calculating errors by restructuring DFGs corresponding to factorization of behavioral descriptions in loop folding.

2 DFG Restructuring and Calculating Error

The prime cause of a calculating error in an operation process is found in round off number treatment due to the restriction of the operation digit of functional units. This round off number treatment is increased more by using a multiplier than by using an adder. However, this rounding error is able to be reduced by factorization of behavioral description.

For example, both of DFGs in Fig.1 are able to obtain the same output, because (b) is the result of factorization of (a). However, the calculating error of (b) is smaller than one of (a). For instance, if a=1.67, b=0.33, and c=0.01 in Fig.1, the calculation result (real value) is 0.02. However, if the figure restriction is 2 decimal places (cut down at 3 decimal places), the calculating result is 0.01 in (a). The result do not agree with the real value because of rounding error.



y=a*c+b*c y=(a+b)*c

(a)before restructuring. (b)after restructuring.

Fig. 1: Restructuring Pattern 1.

On the other hand, the calculating result in(b) is 0.02 under the same input and figure restriction in (a). The result, agrees with the real value, indicates that the rounding error is reduced because of reducing the number of multiplications and putting off multiplications as much as possible by factorization of behavioral description.

Therefore the proposed approach takes the restructuring of a given DFG corresponding to factorize behavioral descriptions in order to reduce calculating errors. Moreover our approach schedules operations in the restructured DFG using loop folding to maximize throughput.

3 Proposed Approach

The proposed approach consists of two phases.

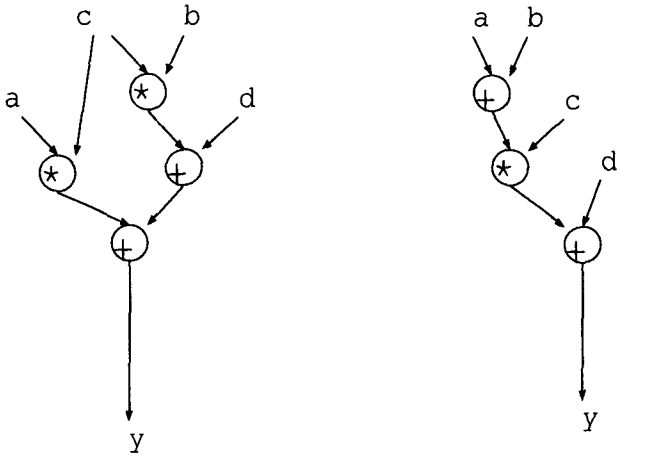
1. Apply the restructuring corresponding to factorization to a given DFG.
2. Schedule operations in the restructured DFG using loop folding.

Particularly, our approach seems to be more effective under the condition that the bit width of each operation is restricted and the figure of value is larger than the figure restriction of operators.

3.1 Restructuring DFG

The object of the DFG restructuring is to factorize to reduce the number of multiplications and put off multiplications as much as possible.

A behavioral description has many restructuring patterns corresponding to the factorization, but it is difficult to search all patterns because of the time complexity of searching restructuring patterns. So the proposed approach applies the restructuring to the two subgraphs as shown Fig.1 and Fig2, because they are the most basic patterns in the actual behavioral descriptions.



a*c+(b*c+d) (a+b)*c+d

(a)Before restructuring. (b)After restructuring.

Fig. 2: DFG Restructuring pattern 2.

Our approach searches restructuring patterns as follows;

for(all nodes){

- *The node, which the number of successors is more than one and two successors (operation node '1' and '2') are multiplications, becomes the node 'c'.*
- *When both node '1' and '2' have only one successor which is an addition, we define these successors node '3' and '4'.*
- *if (node '3' is equal to node '4'), we have searched restructuring pattern1.*
- *else if (node '3' has only one successor which is node '4' or node '4' has only one successor which is node '3'), we have searched restructuring pattern2.*

}

If a restructuring pattern node is on a loop path, the length of the loop is changed by the restructuring DFG. For example, the length of the loop including node "c" or "d" decreases by one node by restructuring. On the other hand, the length of the loop including node "a" increases by one node as shown Fig.3. Because the reduction of the critical loop length corresponds to the maximization of throughput, our restructuring takes the priority of restructuring parts in a DFG as follows;

1. The restructuring parts where the length of the critical loop, effects throughput directly, becomes shorter by the DFG restructuring.

2. The restructuring parts where the length of loops except the critical loop become shorter by the DFG restructuring. These restructuring parts expects that the length of iteration becomes shorter.

3. The restructuring parts where the length of a loop doesn't change by the DFG restructuring.

4. The restructuring parts where the length of loops become longer by the DFG restructuring, if throughput doesn't become smaller than the throughput of the initial DFG.

3.2 Loop Folding scheduling

The proposed approach schedules operations in the restructured DFG by using a loop folding. This loop folding is a method reducing the computation time by the simplification of the evaluation of hardware cost in Theda Fold[2]. To resolve the resource constraint violation, our approach takes the mobility representing the number of assignable steps of each operation in the same manner as List Based scheduling[1], because Theda Fold needs the large computation time for the evaluation of hardware costs. We named this scheduling algorithm *Theda Fold considering Mobility Edition (Theda-ME).*

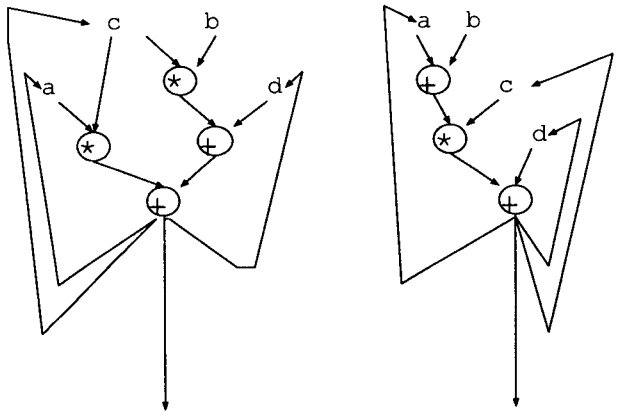


Fig. 3: DFG Restructuring and Loop Length.

4 Experiments

We have implemented the proposed approach in C programming. We named our approach *Error Reduction system in Loop folding Scheduling (ERLS).* We experimented on the second-order IIR filter (IIR)[2] and fifth-order elliptical wave filter (EWF)[3].

To confirm the effect of our approach, *ERLS* was compared with *Theda-ME* (without DFG restructuring).

The experiment program assumed that all operation was executed in one clock cycle.

Table1 shows the results of the number of multiplications and *Latency* corresponding to the interval of output-data. The results of Table1 indicates that the number of multiplications by *ERLS* is smaller than by *Theda-ME.* It is expectable that the calculating errors is reduced by restructuring DFG.

Table2 shows the results of the calculation. The inside of parentheses are the rate of errors. The figure constraint was 3 decimal places (cut down at 4 decimal places). The results of Table2 shows that the *ERLS* really effects to reduce calculating errors.

Table 1: The Num. of Mul. and *Latency.*

(a) IIR

	Theda-ME	ERLS
Num.of Mul.	12	10
Latency	6	6

•resource constraint : two multipliers
two adders

(b)EWF

	Theda-ME	ERLS
Num. of Mul.	12	8
Latency	10	10

•resource constraint : three multipliers
three adders

Table 2: Calculating result.

(a)IIR

values	real	Theda-ME	ERLS
input1	0.04	0.01(75%)	0.03(25%)
input2	24.97	24.83(0.56%)	24.95(0.08%)

(b)EWF

values	real	Theda-ME	ERLS
input3	1.82	1.81(0.55%)	1.82(0%)
input4	16.32	16.24(0.49%)	16.32(0%)

5 Conclusion

This paper proposed a calculating error reduction with graph restructuring in loop folding.

In experimental results, we confirmed that our approach has been able to reduce the calculating errors without increasing of the interval of output-data. So our proposed approach has been effective to reduce propagation of rounding errors without reducing of throughput time in loop folding.

Our future work include applying the proposed approach to DFGs including multi-cycle functional units.

References

[1] Daniel Gajski, Nikil Dutt, Allen Wu, Steve Lin, "HIGH–LEVEL SYNTHESIS Introduction to Chip and System Design, " Kluwer Academic publishers, 1992.

[2] Tsing-Fa Lee, Allen C. -H. Wu, Yong-Long-Lin Daniel D. Gajski, "A Transformation-Based Methord for Loop Folding," IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst., vol.13, no.14, pp.439–450, 1994.

[3] W. F. J. Verhaegh, E. H. L. Aarts, J. H. M. Korst, and P. E. R. Lippens, "Improved force-directed scheduling," IEEE Trans.Comput-Aided Design of Integrated Circuits and Systems, vol.14, no.8, pp.945–960, Aug.1995.

[4] Pierre G.Paulin and John P. Knight, "Force –Directed Scheduling for the Behavioral Sybthesis of ASIC's," IEEE Tans. Comput.–Aided Des, vol.8, no.6, pp.661–679, 1989.

[5] Ganesh Lalshminarayana, Anand Raghunathan, Niraj K. Jha, Sujit Dey, "Power Management in High–Level Synthesis, "IEEE Trans. VLSI Systems, vol.7, no.1, pp.7-15, 1999.

[6] Frederico Buchholz Maciel, Yoshikazu Miyanaga, Koji Tochinai, "Optimizing and Scheduling DSP Programs for High performance VLSI Designs," IEICE Trans. Fundamentals. vol.E–75A, no.10, pp.1191–1201, 1992.

[7] D. A. Lobo and B. M. Pangrle, "Redundant Operater Creation, A Scheduling Optimization Technique, "Proc. of 28th Design Automation Conference, pp.775-778, 1991.