A Scheduling Approach with Component Selection

Katsumi HARASHIMA Hisashi SATOH Daisuke HIRO Toshiro KUTSUWA
Faculty of Engineering, Osaka Institute of Technology
5-16-1, Ohmiya, Ashahi-ku, Osaka, 535-8585 Japan
Phone: +81-6-6954-4305, Fax: +81-6-6957-2136
E-mail: harasima@elc.oit.ac.jp

Abstract: The reduction of chip area and delay is important purpose of Scheduling in High-Level Synthesis. This paper presents a scheduling approach with component selection. After obtaining a initial schedule taking only single-functional units, the component selection of our approach attempts the reduction of chip area and/or delay by the selection more suitable components in a component library using Simulated Annealing.

1 Introduction

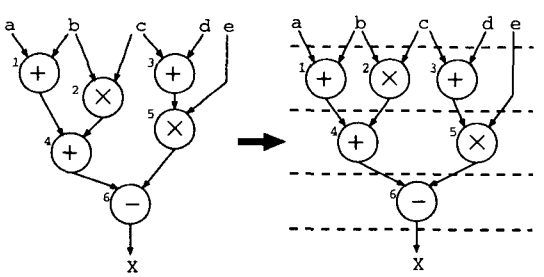
High-Level Synthesis(HLS) is a technology that generates a register transfer description automatically from a behavioral description. Scheduling in HLS is a important phase because it influences chip delay and area[1]. Scheduling assigns each operation to a particular control step in order to minimize chip area and delay exploiting parallelism between operations. Because of these contrary purposes, it generally considers one as restrictions and minimizes the other. In order to simplify the scheduling problem, most of conventional approaches take the component restriction that each type of operation can be performed by one and only one type of functional units[2]–[5]. However this component restriction is not realistic. In practice, designers select suitable components in the circuit component library to design LSIs having a different characteristic.

In this paper, we propose a scheduling approach with component selection in order to minimize chip area and delay. The proposed approach obtains a initial schedule by the ASAP scheduling with the conventional component restriction, and then replaces current components by more suitable ones in order to reduce chip area and/or delay using Simulated Annealing (SA). SA is a control strategy that applies the annealing process in physics to combinational optimization to move out a locally optimal solution and on toward a globally optimal solution. Our approach can reduce chip area and delay simultaneously by using SA for the replacement of components, called the component selection.

2 Scheduling

The scheduling problem requires a data flow graph (DFG) corresponding to the behavioral description. A DFG is a directed acyclic graph, where each node represents an operation in the behavioral description. The edge from node O_1 to node O_2 exists if operation O_2 consumes the result produced by operation O_1.

The task of scheduling decides the computation order of all operations under the condition that it dose not break the data dependencies. Figure 1 shows an optimal scheduling without the constraint of the number of available functional units. But the fact that a multiplier is slower than an adder equalizes the scheduling result in Fig.1 to the result in Fig.2. As a result, adders idle during part of the clock cycle and so are underutilized.



$$X = \{(a+b)+(b\times c)\}-\{(c+d)\times e\}$$

Fig. 1: Scheduling.

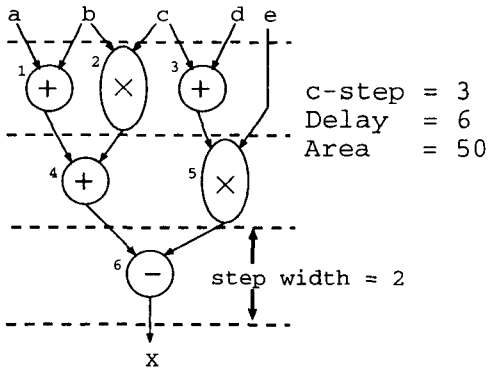


Fig. 2: Consideration of delay.

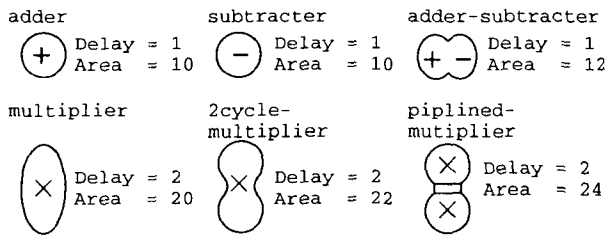


Fig. 3: Component library.

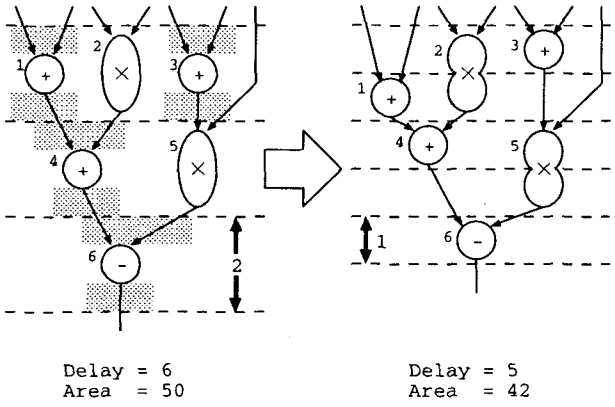


Fig. 4: Replacement 1.

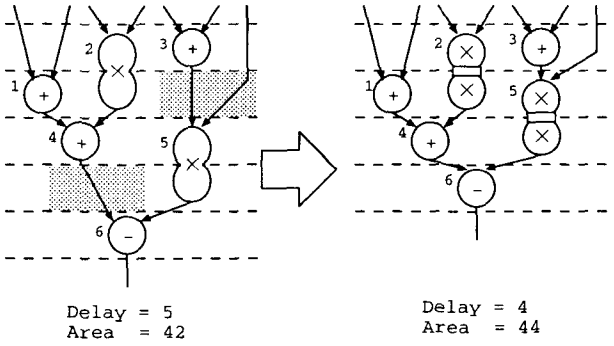


Fig. 5: Replacement 2.

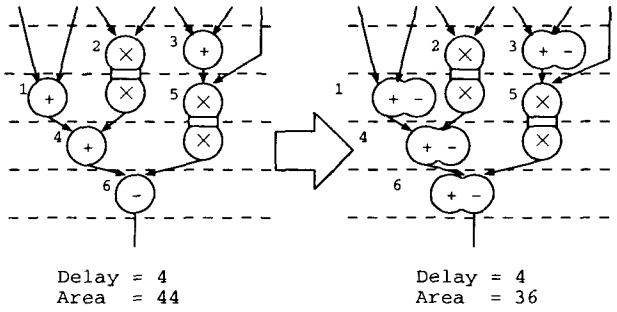


Fig. 6: Replacement 3.

3 Scheduling Algorithm

The proposed approach obtains a initial scheduling by using the ASAP scheduling, which assigns operations into the earliest control steps within which the operations are to be scheduled, and then replaces current functional units by more suitable ones in order to reduce chip area and delay with SA.

3.1 Component Selection

In the component selection, our approach replaces current functional units by units having the effect of the reduction of chip area and/or delay in component libraries. A component library contains multiple types of functional units, with different characteristics. In our approach, available functional units are single-functional (e.g., adder, subtracter, multiplier), multi-functional (e.g., adder-subtracter, ALU), multi-cycle, and pipelined units as shown in Fig.3. Since the ASAP scheduling in our approach uses only single-functional units, our replacement takes three techniques.

1. Replace with multi-cycle units
   This replacement changes a single-cycle unit to a multi-cycle unit to reduce chip delay. For example, two adders can execute during a 2-cycle multiplier and so take little idle time(Fig. 4).
2. Replace with pipelined units
   This replacement changes a multi-cycle unit to a pipelined one to reduce chip area and delay. For example, two multiplication can share the same 2-stage pipelined multiplier, which the two multiplication execute at two successive control steps(Fig. 5).
3. Replace with multi-functional units
   This replacement changes single-functional units to a multi-functional unit to reduce chip area. For example, an adder and a subtracter, execute at different control steps, are changed to an adder-subtracter(Fig. 6).

Since the number of combinations of operations and functional units grows rapidly with the number of nodes in a given DFG, heuristic approaches have difficulty in finding the optimal solution. Therefore, our approach replaces functional units using SA.

3.2 Introduction of Simulated Annealing

Table 1 shows the correspondence between SA and the component selection.

Table 1: Correspondence between SA and the component selection.

SA	component selection
system state	circuit state
energy	hardware cost
change of state	replacement of components
temperature	control parameters
equilibrium state	solution

In the proposed approach, the gain $G(X)$ of the hardware cost in the circuit state X represents the reduction rate of hardware cost compared with the hardware cost of the initial scheduling, and is formulated as follows:

$$\begin{aligned} G(X) &= P_a \left(A(X) / A_{init} \right) \\ &\quad + P_u \{ 1 - (U(X) / U_{init}) \} \\ &\quad + P_s \left(S(X) / S_{init} \right) \\ &\quad + P_d \left(D(X) / D_{init} \right) \end{aligned} \quad (1)$$

where $A(X)$ is the sum of area of functional units, $U(X)$ is the average operating ratio of all functional units, $S(X)$ is the number of control steps, $D(X)$ is the execution delay, A_{init}, U_{init}, S_{init}, and D_{init} are the initial values of $A(X)$, $U(X)$, $S(X)$, and $D(X)$ respectively, and P_a, P_u, P_s, and P_d are alternative parameters.

If the new state is worst than the old state, SA avoids locally minimal solutions by accepting the new state with a probability determined by control parameters and a random number. But SA can not avoid locally minimal solutions perfectly because it generates the new state by replacing functional unit randomly. Our approach improve the capacity to find the optimal solution by taking a look-ahead scheme. The a look-ahead scheme selects the replacement with the best gain among m next state candidates. Each candidate is obtained by repeating our functional unit replacement by SA n times.

Figure 7 shows the result of the application of our component selection to the schedule in Fig. 2. Our component selection has been able to optimize both of chip delay and area.

4 Experiments

Our approach was tested on five DFGs generated randomly. Tables 2 – 6 show the experimental results. In each table, Init is the initial scheduling

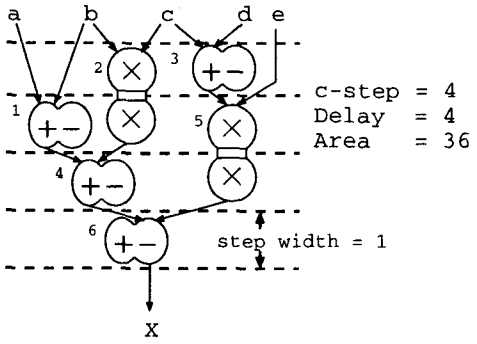


Fig. 7: Result of the component selection.

Table 2: Result for a DFG with 20 operations.

	A	#U	U	S	D	CPU [s]
Init	3100	8	17.08	9	90	—
Iter	2920	6	26.20	9	90	0.35
Ours	2740	2	96.43	14	77	0.81

by using the ASAP scheduling, Iter is a conventional iterative refinement method and Ours is the proposed approach. A, U, S and D are the final value of $A(X)$, $U(X)$, $S(X)$ and $D(X)$ respectively, #U is the number of functional units, and CPU is the computational time.

In experiments, we took following alternative parameters.

$$P_a = (Node\_Num/100)^{0.75} \quad (2)$$

$$P_d = 1/P_a \quad (3)$$

if $P_A > P_d$

$$P_s = P_d/4 \quad (4)$$

$$P_u = P_d/2 \quad (5)$$

if $P_a < P_d$

$$P_s = P_a/4 \quad (6)$$

$$P_u = P_a/2 \quad (7)$$

where $Node\_Num$ is the number of operations in each DFG.

We confirmed that the solution in Fig. 2 was optimal by searching all replacements. Solutions for other four DFGs improved the operating ratio, area, delay and the number of functional units compared with the iterative method. Especially, the operation ratio was improved about double. This result is effective to reduce chip area because few functioan units and signal lines idle.

Table 3: Result for a DFG with 50 operations.

	A	#U	U	S	D	CPU [s]
Init	6650	16	13.84	14	140	—
Iter	3280	7	36.12	14	140	1.97
Ours	2130	3	80.47	20	120	7.36

Table 4: Result for a DFG with 100 operations.

	A	#U	U	S	D	CPU [s]
Init	14440	22	14.57	18	180	—
Iter	4300	11	29.14	18	180	8.70
Ours	4050	5	71.26	27	162	11.44

Table 5: Result1 for a DFG with 500 operations.

	A	#U	U	S	D	CPU [s]
Init	28270	42	14.38	48	480	—
Iter	4090	10	42.02	68	748	273.30
Ours	4020	8	85.03	72	396	338.51

Table 6: Results2 for a DFG with 500 operations.

	A	#U	U	S	D	CPU [s]
Init	35210	49	12.25	49	490	—
Iter	4870	7	76.78	88	528	187.76
Ours	3330	5	88.69	107	642	389.71

In all results, our approach required the larger number of control steps than the iterative method. However execution delays in proposed approach were shorter than Iter because the clock cycle became shorter by using pipelined and multicycle units effectively in our component selection.

Although the computational times became longer than the iterative method, they are reasonable.

5 Conclusion

This paper has proposed a scheduling approach with component selection. After obtaining a initial scheduling by the ASAP scheduling, the component selection minimizes chip area and delay by using SA in the component selection because SA can avoid falling down locally optimal solutions. As experimental results, we have confirmed that our approach can obtain near-optimal solutions.

Our future work will include applying the proposed approach to practical data and then confirm the availability of it.

References

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

[2] M.K.Dhodhi, F.H.Hielscher, R.H.Storer and J. Bhasker, "Datapath Synthesis Using a Problem Space Genatic Algorithm," IEEE Trans. Comput.-Aided Design of Integrated Circuits ans System, Vol.14, No.8, pp.934–944, 1995.

[3] W.F.J.Verhaegh, P.E.R.Lippens, E.H.L.Arts, J.H.M.Korst, J.L.van Meerbergen and A.van der Werf,"Improved Force-Directed Scheduling in High-Troughput Digital Signal Processing," IEEE Trans. Comput.-Aided Design of Integrated Circuits ans System, Vol.14, No.8, pp.945–960, 1995.

[4] Kenji Ohmori, "High-level Synthesis Using A Genetic Algorithm," Trans. of IEICE, Vol.J81-A No.5 pp.854-862, 1998.

[5] Sanghum Park and Kiyoung Choi, "Performance-Driven Scheduling with Bit-Level Chaining," Proc 36th Design Automation Conf., pp.286–291, 1999.