An Efficient Hardware Allocation Algorithm for Optical Hardware Architecture Design

Kyung-Min Eom[1], Dal-Hwan Yoon[2], Chi-Ho Lin[3]
[1 3] School of Computer Science, Semyung University
[2] Dept. of Electronic Engineering, Semyung University
Tel: +82-43-649-1272, Fax: +82-43-649-1770
E-mail: [1]eom4544@hanmail.net, [2] yoondh@semyung.ac.kr, [3]ich410@semyung.ac.kr

Abstract: This paper proposes an efficient hardware allocation algorithm for optical hardware architecture design.

The proposed algorithm works on scheduled input graph and allocates binds function-al units, registers and interconnections by considering interdependency between operations and memory in elements in each control step, in order to share registers and interconnections connected to functional units, as much as possible. Also, the register allocation is especially executes the allocation optimal using graph coloring techniques. Therefore the overall resource is reduced.

The effectiveness of the proposed algorithm has been proven by the experiment with the benchmark examples.

1. Introduction

It is target purpose of CAD technology that a process from behavioral description of designed IC chip to design automation for chip manufacturing. The modern logic design (from RT level circuit description to gate level circuit design) or the layout design (form the technology library mapped each circuit element to placement and routing) is commonly used. But the study on the high level synthesis that documentation feature of design flow, according to integration and complexity of Hardware Architecture be shorten design time, to reduce debugging time and error of various design automation for evaluation chip performance design of the beginning stages, is lacking. The definition of high-level synthesis was consisted of scheduling, allocation; binding from the behavioral description of designed to create structure of RT register transfer level for limiting constraints and satisfied target function. The scheduling consist of assigning behavioral description each operation to control step. But it had developed algorithms solution of the limited application field for very scheduling process is considerable items conditional branch, pipeline, loop and so on.[1-4]

The allocation is assigned so that minimize area of implement hardware, to operation as functional unit, to variable as register, between operation and register as interconnection assigned to bus and multiplexer. The typical method is greedy allocation, left edge algorithm, clique partitioning and so on. The greedy allocation which is assigned hardware resource for executive variable and operation each at the time interval. The clique partitioning which is applied allocation of operation and memory as the approach method for using the graph theory. As the existed method of a hardware allocation, HAL[5] system is together executes allocation of functional unit and scheduling to suppose one each type of functional unit, assigned allocation and binding to register and interconnection to use clique partitioning method. Splicer system[6] is together executed scheduling number of function unit which fixed the states in advance, the number of register minimized interconnection using the method of branch and bound. But this system cannot to obtain optimal design. Also, REAL[7] system allocates minimum register using the left-edge algorithm in this case not considering mutual exclusion. But does not consider influence interconnection, not deal with the allocation of function unit and interconnection. The existed methods that the first, the number and type of the register and functional unit is fixed in advance, the second, it execute to separate the allocation and binding. [8-9]

Consequently this paper proposes an efficient hardware allocation algorithm for optical hardware architecture design. The proposed algorithm works on the scheduled input graph and simultaneously allocates and binds registers, functional units and interconnections in stages by considering interdependency between operations and storage element in each control step, in order to share registers and interconnections that are connected to functional units, as much as possible.

The structure of this paper is introduction of the first section, section 2 describes the proposed an efficient hardware allocation algorithm for optical hardware architecture design, section 3 describes experimental result in our proposed algorithm, and finally section 4 gives conclusion.

2. The proposed an efficient hardware allocation algorithm

In this paper, the proposed an efficient hardware resource allocation algorithm is shown in figure 1. The input works on the scheduled input graph and functional units calculated in order to allocate and bind for mobility of all operation at preprocessing. The mobility of operation is computed by investigating data dependency. From the first control step allocate register, functional unit, and interconnection in stages, after calculating mobility of operation. At this point, providing that allocation and binding of functional units finished, the mobility of existing operation modify at the next control steps. The interconnection merging executes, after allocation and binding, as control step on the whole.

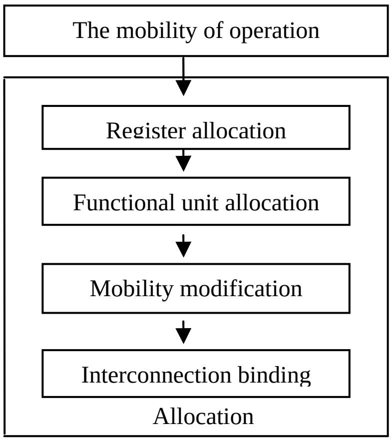


Figure 1. The overall flow of the an efficient hardware resource allocation algorithm

2.1 The register allocation

The register allocation and binding is executed in stages each control step, alike the next description.
The first register classifies to allocate type. Namely, variable or constant, before the control step executed output of functional unit, classify. The second, it allocate that a register, according to classify type. In this case variable, if the first control step, new register assign and if the next control step, register allocate that reused before the control step.

Namely, The overlapped register allocation but existing another control step executes optimal register allocation using graph coloring techniques. After the life-time composed according to arrange input created graph. When it suppose usable register number is K, if the node don't exist with degree(n) < k (n: node, k: usable register number) insert node of stack in position in stead of the spill. The coloring execute optimal coloring that node don't coloring after it suppose that color is able to use at the stack when the node pop at stack, if color not useable. The coloring algorithm shown in figure 2. In this case constant, it excluded register allocation. In this case of the output of functional units that was performed at the previous control step, investigate whether it is the input of the other operation, allocation at register after considering the types of the functional units and the type of operation receiving input. If it is not the input of other operation, allocation at register after considering the type of the function operator.

```
if(node) {
      color_stack_pop( );  /*  Pop stack  */
      if(degree(n) > k) {
         Non_coloring(  );   /* Not coloring */
         Spill_code(  );  /* Insert spill code  */
      else
         Coloring(  );   /* Coloring */
      }
}
```

Figure 2. The coloring algorithm

If the loop exists, allocate such as Figure 3, the register that is used at the beginning and ending of a loop. That is, each variable V1, V2, V3 is allocated as register R1, R2, R3 the first control step.
At the same time they are allocated as register R1, R2, R3 which or the same register at the last control step.

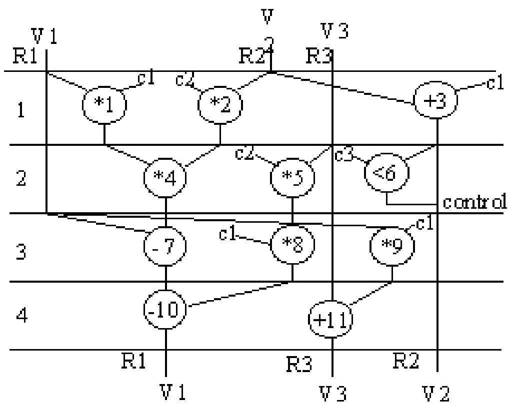


Figure 3. Register allocation for loops

2.2 The functional unit allocation

After performing of allocation and binding of register, performance the allocation of the functional unit about each operation which is being, now, at the control step. That is, to choose the functional unit, in the cell library the functional unit, which are satisfied with the computed performance time of each computed of operation and has the smallest area, investigate the five variable of self-distributed number, relative distributed number, self-fixed num ber, relative fixed number, and self mobility that will allocate or bind the functional unit. First, self-distributed number is at the control step such as operation which is going to allocate the functional units and represents the number of operation which has the same type. Relative distributed number represents operation which is going to allocate the functional units and maximum number of operation which is at the other control step, has the same type.

Also, Self-fixed number is the maximum of operation of which mobility is zero when it is investigated the mobility of operation which exist at the same control step and has the same type, Relative fixed number is the number of maximum of which mobility is zero per a control step when it is investigated the mobility of operation which has the same type and is at the other control step with the operation that is going to allocate functional units. Self mobility is the mobility of operation that is going to allocate functional units.

In figure 4. the self-distributed number of multiplication operation which is at the first control step is one.

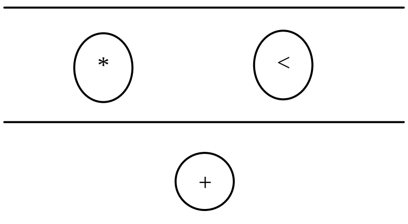


Figure 4. A example of self-distributed number

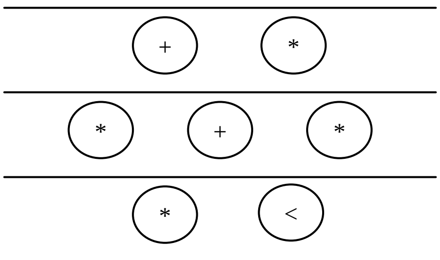


Figure 5. A example of relative distributed number

An example of relative distributed number shown in figure 5. The relative distributed number, if multiplication operation which is at the first control step is determined as follows. The number of multiplication operation which is at the second control step is two, the number of multiplication operation which is the third control step is one, that has the maximum value, so the relatives distributed number of multiplication is two. An example of self fixed number shown in figure 6. It is possible to execute at the second control step. The multiplication operation which is at the first control step can be also performed at the control step, so the mobility is one. Therefore the self fixed number is zero.

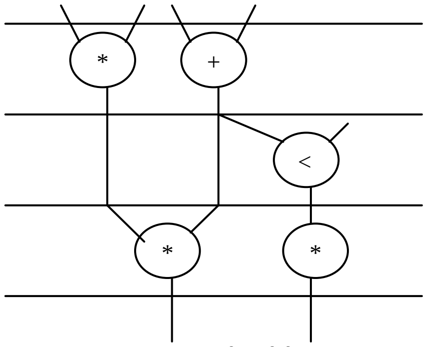
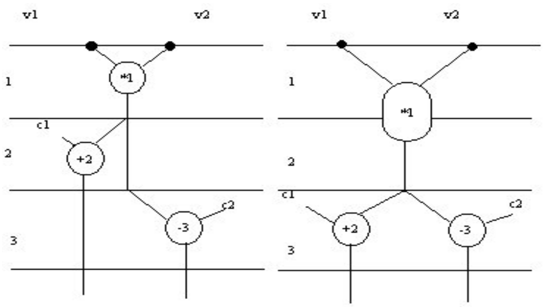


Figure 6. A example of self-fixed number

2.3 A step modification the mobility

When you allocate the functional unit for operation, in the case of using the multi-cycling which uses many control step, the mobility arbitrate for all operation of dependence the operation. Plural control step is the same with the delay correction time on the library.

For example, figure 7(a). is *1 allocates and binds functional units for using multiple control step, self distributed number 1 and self mobility 2, self fixed number, relative distributed number, relative fix number, all 0. There by the +2 of input received output of *1 is mobility conversed from 1 to 0. The result of allocation and binding shown in figure 7(b).



(a) (b)

Figure 7. A example of modification the mobility
(a) Before the modification of mobility
(b) After the modification of mobility

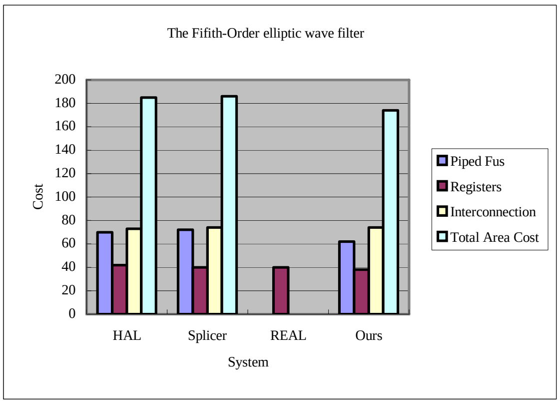
2.4 The binding of interconnection

Perform the allocation of interconnection after performing the allocation of functional units. In this case first control step, allocate new multiplexer for each operation. Second, from control step, investigate the types of functional units and input, and then find out the same type as much as possible finally allocate the multiplexer. Third, investigate the input number of multiplexer. If, there is one, omit multiplexer, or not investigate the control step of each multiplexer. If the input value is the same, it will be merged even through the control step is duplicated or not. At this time, the multiplexer which has been merged stand face to face with bus. As an example, seeing the figure 6, we can find out that *5 which is at the second control step and *2 which is at the first control step have been allocate and bind for the same functional unit (FU2). Therefore, the interconnection of *5 can allocate and bind as the same interconnection of *2.

3. The result of experiments

This paper executed allocation algorithm result is compared by HAL[5] result, for exactly comparison, using HAL, Splicer[6] application extraction result is received input. The HAL, Splicer, REAL[7] comparative result of area cost of the fifth elliptic wave filter to adopt as the standard benchmark model for High-Level synthesis Workshop as benchmark model shown in Table 1. The HAL system is the piped functional units area cost was reduced 11.1%, Also register area cost was reduced 9.3% , consequently total area cost is reduced. The Splicer and REAL system is the same cost ratio register like the differential equation. The total area cost reduced 5.8%.

Table 1. The benchmark experiment result for fifth -order elliptic wave filter



4. Conclusion

This paper have showed a new algorithm that perform a hardware resource allocation and binding for optical hardware architecture. A hardware resource allocation algorithm performed with the characters as follows. First, from control step, allocate registers and functional units by stages and then perform interconnection merge after performing interconnection binding.

Finally, the hardware cost functional unit values was shows effectiveness, which minimum for the ultimate purpose of high-level synthesis techniques. Also, after this study project will precede the study for the anticipation and estimation which based on a simultaneous hardware resource allocation and binding algorithm for optical SOC design.

Reference

[1] Breuer, M. A., *Digital System Design Automation*, Computer Science Press, Inc. 1975.

[2] Rubin, S. M., Computer Aided for HARDWARE ARCHITECTURE Design, Addison_Wesley.

[3] Shiva, S. G., Jan., "*Automatic Hardware Synthesis*," Proceedings of the IEEE, Vol. 71(1), p.76-87, 1983.

[4] Gajski, Daniel D., 1988, *Silicon Compilation*, Addison-Wesley

[5] Brayton, R. K. Sangiovanni-Vincentelli, A. L. and G. D. hatchtel, "*Multi-level Logic Synthesis*," Proceedings of the IEEE, Vol.78(2), p.264-300. Feb. 1990.

[6] Kuh, E. S. and Ohtuski, T., "*Recent Advance in HARDWARE ARCHITECTURE Layout*," Proceedings of the IEEE, Vol. 78(2), Feb 1990.

[7] McFarland, M. C., Paker,A. C. and Camposano, R., "*The High-Level Synthesis of the Digital System*," Proceedings of the IEEE, Vol. 78(2), p.301-318, Feb. 1990.

[8] Hitchcock, C. Y., and D. E. Thomas, "*A Method for Automatic DataPath synthesis*," Proc. of the 20th Design Automatic Conference(DAC), p.484-489. 1983.

[9] Camposano, R., "*From Behavior to Structure: High-Level Synthesis*," IEEE Design & Test of Computer", p.8-19, Oct. 1990.

[10] Daniel D. Gajski, Nikil D. Dutt, and Allen C-H Wu, "*High -Level Synthesis: introduction to chip and system design*," p.272 -283, 1992.