A Resource-constrained Scheduling Algorithm for High-level Synthesis

Ho-Jeong Song, Jae-Jin Lee, In-Jae Hwang*, Gi-Yong Song

Dept. of Computer Engineering   Dept. of Computer Education*

Chungbuk National University, Cheongju Chungbuk 361-763 Korea

Tel : +82-43-261-2452   Fax : +82-43-262-2449

E-mail : gysong@chungbuk.ac.kr

Abstract : Scheduling is assigning each operation in a control/data flow graph(CDFG) to a specific control step. It directly influences the performance of the hardware synthesized. In this paper, we propose an efficient resource-constrained scheduling algorithm assuming that only available silicon area is given. We performed the experiment to evaluate its performance. The results show that our algorithm find the solution with shorter scheduling length compared to the existing methods.

1. Introduction

Scheduling for digital system synthesis is assigning each operation in a control/data flow graph(CDFG) to a specific control step without violating precedence relation. It is one of the most important tasks due to its direct influence on the performance of the hardware synthesized. In general, scheduling problems can be grouped into two classes; time-constrained scheduling and resource-constrained scheduling. In time-constrained scheduling, a fixed control step length is given, and the objective is minimizing the cost of hardware by minimizing the number of functional units of each operation. Time-constrained scheduling is important for designs of real time systems where sampled data arrive continuously at a fixed rate.

As opposed to time-constrained scheduling, resource-constrained scheduling is required when the limitation is imposed on the silicon area. The goal here is producing a design with the best possible performance while meeting the given area constraint. Scheduling an operation can be delayed by the limited amount of resources, even if all the preceding operations are already scheduled. There are many resource-constrained scheduling algorithms existing in the literature. Most of them assumed that the constraint is given in terms of the number of functional units of each type[2]. However, what is given in the real design problem is the available silicon area, and from which it is very difficult to determine the number functional units of each type that can accommodate the shortest scheduling length.

In this paper, we propose an efficient resource-constrained scheduling algorithm assuming that only available silicon area is given. Our algorithm first analyzes the given CDFG to determine the number of functional units of each type, then assigns each operation to a control step while satisfying the constraints. It also tries to improve the solution iteratively by adjusting the number of functional units using the results collected from the previous scheduling.

2. Proposed approach

In the approach we propose, we determine the number of functional units of each type using simple scheduling algorithms such as ASAP and ALAP scheduling[1]. ASAP scheduling algorithm assigns a value e_i to each node v_i of a CDFG. e_i represents the earliest possible control step that v_i can be scheduled with the given precedence constraints. The control step length E obtained by ASAP scheduling is the shortest possible scheduling length provided that unlimited number of functional units are available.

After ASAP scheduling, we apply ALAP scheduling to the given CDFG with the control step length E to obtain l_i value for each node v_i. l_i represents the latest possible control step that v_i can be scheduled without exceeding the scheduling length E. Hence, the values e_i and l_i provide the range of control steps for the operation represented by the node v_i. Figure 1. show the results of applying ASAP and ALAP scheduling to a given simple CDFG.

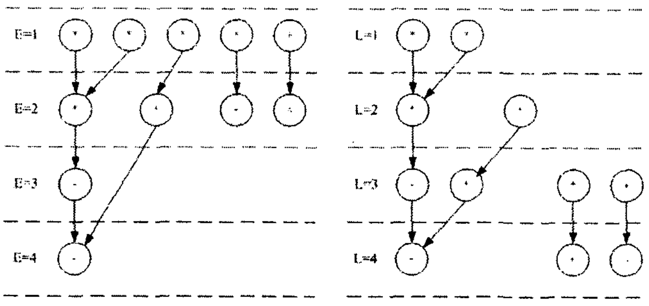

Figure 1. ASAP & ALAP scheduling

A functional unit should be available between the control step e_i and l_i so that v_i can be scheduled in that range. If many operations of the same type have the same range, we need many functional units of that type to perform those operations. In our proposed approach, we use the above information to determine the initial number of functional units of each type. For each operation type t_k, we compute $p_{t_k}^j$ as follows.

$$p_{t_k}^j = \sum_{v_i} \frac{1}{l_i - e_i + 1} \quad \text{where } v_i \text{ is of type } t_k \text{ and } e_i \leq j \leq l_i$$

$p_{t_k}^j$ is the estimated number of functional units of type t_k at control step j. We take the maximum of $p_{t_k}^j$ to obtain the number of functional units of type t_k. That is, $p_{t_k} = \max_j p_{t_k}$. The p_{t_k} values are used as the proportional distribution of the number of functional units of each type. The actual number of functional units n_{t_k} is obtained using the above distribution, available silicon area, and the silicon area required for implementing each functional unit. Since n_{t_k} is an integer value, it is possible that a portion of silicon area is not used because it is too small for any functional unit. We trace the unused area in the remaining procedure to maximize the utilization of silicon area.

After obtaining n_{t_k} values, we use list scheduling to assign each operation to a control step. In the list scheduling, we try to identify the operation type, such that increasing the number of functional units of that type may shorten the scheduling length. This is done by counting the number of operations of each type that cannot be scheduled due to the lack of functional units. We increase the number of functional units of that type by decreasing the numbers of functional units of other types. Then we apply the list scheduling again with the updated number of functional units. We repeatedly apply the same procedure until no more improvement is possible. In the following section, we present the proposed approach in detail.

3. A scheduling algorithm

We use the following notations in description of the proposed scheduling algorithm.

G	: Given Control/Data Flow Graph
o_i	: Operation represented by node v_i in G
A	: Total silicon area
t_k	: Functional unit type
a_{t_k}	: Required silicon area to implement functional unit of type t_k
$p_{t_k}^j$	: Estimated number of functional units of type t_k at control step j.
n_{t_k}	: Number of functional units of type t_k
R	: Remaining silicon area
C_{step}	: Control step length

The following procedure compute a_{t_k} values using the method explained the in the previous section.

```
Procedure COMPUTE_FUNC_UNITS ($G, A, a_{t_k}$)

    Call ASAP ($G$)
    Call ALAP ($G$)
    for each $o_i$ do
        for each $j$, $e_i \leq j \leq l_i$ do
        $P^j$ (type of $o_i$) = $P^j$ (type of $o_i$) + 1 / ($l_i$-$e_i$+1)
        endfor
    endfor
    for each $t_k$ do
        $p_{t_k} = \max_j P^j_{t_k}$
    endfor
    for each $t_k$ do
        $q_{t_k} = \frac{P_{t_k} a_{t_k}}{\sum_{t_k} P_{t_k} a_{t_k}} \cdot A$
        $N_{t_k} = \left\lfloor \frac{q_{t_k}}{a_{t_k}} \right\rfloor$
    endfor
    $R = A - \sum_{t_k} N_{t_k} a_{t_k}$
end COMPUTE_FUNC_UNITS
```

After obtaining n_{t_k} values, the following procedure is called to assign each operation to a control step. It is essentially a list scheduling, but we count the number of operations of each type that cannot be scheduled due to the lack of functional units in each iteration. These numbers are used in the procedure REALLOCATE to adjust the number of functional units to obtain the better scheduling. The procedure INSERT_READY_OPS scans the set of nodes in G to determine if any of the operations are ready. It deletes each ready node from the set and appends it to one of the priority lists based on its operation type. Procedure SCHEDULE_OP schedule the given operation at the given control step. Procedure DELETE deletes the given operation from the specified list.

```
Procedure SCHEDULING ($G$, $n_{t_k}$)

    Prev_S_length = $\infty$
    repeat
        INSERT_READY_OPS($G$, $Plist_1$, $Plist_2$, …, $Plist_{t_m}$)
        $C_{step}$ = 0
        while (($Plist_1$=Ø) or ($Plist_2$=Ø) or $Plist_{t_m}$ = Ø)) do
            $C_{step}$ = $C_{step}$+1
            for $k$=1 to $m$ do
                for $funi$=1 to $n_{t_k}$ do
                    if $Plist_{t_k}$ = Ø then
                        SCHEDULE_OP(Scurrent,FIRST($Plist_{t_m}$), $C_{step}$)
                        $Plist_{t_m}$ =DELETE($Plist_{t_m}$, FIRST($Plist_{t_m}$))
                    endif
                endfor
                $R_{OP_{t_k}} = R_{OP_{t_k}} + 1$
            endfor
            INSERT_READY_OPS($G$, $Plist_1$, $Plist_2$, …, $Plist_{t_m}$)
        endwhile
        $R\_OP_{t_{\min}} = \underset{t_k}{Min}(R\_OP_{t_k})$
        $R\_OP_{t_{\max}} = \underset{t_k}{Max}(R\_OP_{t_k})$
        REALLOCATE($R$, $t_{min}$, $t_{max}$)
        $S_{prev}$ = $S_{current}$
    until ($C_{step}$  Prev_S_length)
    $S$ = $S_{prev}$
End SCHEDULING
```

```
Procedure REALLOCATE($R$, $t_{min}$, $t_{max}$)
    if $R \geq R \geq a_{t_{\max}}$ then
        $R$=$R$- $R \geq a_{t_{\max}}$
    else
        temp= $a_{t_{\max}} - R$
        R = 0
        repeat
            temp = temp - $a_{t_{\max}}$
            $n_{t_{\min}}$ = $n_{t_{\min}}$ - 1
        until (temp ≤0)
        $n_{t_{\max}}$ = $n_{t_{\max}}$ + 1
end REALLOCATE
```

4. Performance evaluation

We performed the experiment to evaluate the performance of our scheduling algorithm. The proposed scheduling algorithm was applied to the CDFG of fifth-order wave digital elliptic filter[3]. This example contains 26 additions and 8 multiplications. The scheduling length obtained by ASAP scheduling was 33, which is the shortest possible length with unlimited number of functional units. We assumed that a multiplication unit takes the twice as much silicon area as an addition unit. We calculated the equal number of multiplication and addition units that can be accommodated in a given silicon area. We applied the list scheduling algorithm with the above functional unit numbers and obtained the scheduling length 38. We also applied our scheduling algorithm to the same example and obtained the shorter scheduling length 35.

Table 1. Performance analysis of the digital elliptic filter for the case A= 8

Algorithm	num of +	num of *	Schedule length
List Schedule	2	2	38
	2	3	38
Our Schedule	4	2	35

5. Conclusions

In this paper, we propose an efficient resource-constrained scheduling algorithm assuming that only available silicon area is given. The algorithm utilizes the distributions of different operations in the control/data flow graph to determine the number of functional units. The operations are scheduled using well-known scheduling algorithm, and the result is iteratively improved. We performed the experiment to evaluate its performance. The results show that our algorithm find the solution with shorter scheduling length compared to the existing methods.

References

[1] Daniel D.Gajski, Nikil D.Dutt, *High-Level Synthesis Introduction to Chip and System Design*, Kluwer Academic Publishers

[2] C.T. Hwang and Y.C. Hsu, "Zone Scheduling", *IEEE Transactions on Computer-Aided Design of Integrated Circuites and Systems*, vol.12, *no.7, pp.926-934, July 1993.*

[3] S.Y. Kung, H.J. Whitehouse, and T. Kailath, "VLSI and Modern Signal Processing", Englewood Cliffs, NJ: Prentice Hall, pp.258-264, 1985.