Heuristic-Based Scheduling Algorithm for High Level Synthesis

Gulam Mohamed, Han-Ngee Tan & Chew-Lye Chng
School of Electrical & Electronic Engineering
Nanyang Technological University
Singapore 2263
Republic of Singapore
e-mail : gmohamed@ntu.ac.sg

Abstract - **A new scheduling algorithm is proposed which uses a combination of resource utilization chart, a heuristic algorithm to estimate the minimum number of hardware units based on operator mobilities and list-scheduling technique to achieve fast and near optimal schedules. The schedule time of this algorithm is almost independent of the length of mobilities of operators as can be seen from the benchmark example (fifth order digital elliptical wave filter) presented when the cycle time was increased from 17 to 18 and then to 21 cycles. It is implemented in C on a SUN3/60 workstation.**

1 Introduction

High level synthesis of digital system involves mapping of an abstract behavioral or algorithmic level specification to a register-transfer-level structure while satisfying a set of constraints. The system will normally output a datapath structure which implements the specification together with a controller unit. The major tasks involved are i) extraction of timing and data precedence relationship from the input ii) scheduling of operators into timesteps iii) allocation of hardware resources like functional units, storage devices and interconnects and iv) creation of the control unit[1]. Among these tasks, operator scheduling in (ii) has been acknowledged to be one of the most crucial step[2]. Decisions made in this step will have direct consequence on the performance and cost of the design in VLSI implementation.

We consider in this paper the problem of scheduling under time constraint, i.e., finding the cheapest schedule without exceeding the given number of time steps. A heuristic based algorithm is proposed which tracks hardware usage, estimates the minimum number of hardware units required based on operator mobilities and uses list-scheduling technique to place operators to timeslots while minimizing the number of hardware functional units, lifetimes of variables and additional multiplexer costs. Our proposed algorithm is able to place operators into timeslots in almost linear time when their mobilities were increased as can be seen in the benchmark example presented. In this paper we consider the scheduling algorithm which minimizes the total number of hardware functional units only. The proposed algorithm in its fullest implementation includes storage device and multiplexer costs considerations. This is not discussed here due to space limitation but can be found in [3].

This paper is organized as follows. Section 2 describes briefly previous related works. The scheduling problem is formulated in Section 3. Section 4 describes our Nanyang Technological University Scheduling System (NTUSS) followed by a brief description of the basic scheduling

algorithm in Section 5. Section 6 gives results for a benchmark example and finally Section 7 concludes this paper.

2 Previous Works

A. C. Parker et al. introduced the notion of freedom-based scheduling in MAHA[4]. Operators with the smallest degree of freedom (bounded by ASAP and ALAP times) are given the highest priority when considered for scheduling. In Force-Directed Scheduling[5], P. G. Paulin and J. P. Knight use 'force' to determine the suitability of an operator to a timestep. The 'force' value favors a balanced distribution of operators over all the control steps. On the other hand, the suitability of placing an operator to a control step is determined by a selection function in the system described in [6] by Park In-Cheol and Kyung Chong-Min. Similarly, their selection function is based on balancing the distribution of operators in each control step. Hwang C. T. et al.[7] use integer linear programming model to find the optimal operator-timeslot combination while minimizing a cost function which includes hardware costs. In most of these systems the time taken to schedule increases tremendously with increasing length of mobilities of operators.

3 Problem Formulation

For a given Acyclic Directed Dataflow graph, Critical Path analysis is used to determine the possible timesteps an operator can occupy based on their given processing times and data precedence relation. The longest path from the input to the output represents the critical path and the latest time the last operator can start will place an upper bound on the maximum timesteps the dataflow graph takes to produce its outputs. Every operator will have to be restricted to within their ASAP and ALAP times if the last operator in the dataflow graph were to start its operation within the upper bound. Operators having their ASAP times equal to their ALAP times are in the critical path and are scheduled to their respective ASAP times. We use a combination of resource utilization chart, heuristic algorithm to estimate the minimum number of hardware units required and list-scheduling technique[8] to place the rest of the operators to one of the timepoints within their ASAP and ALAP times such that :

(i) data precedence relationship as defined in the input dataflow graph is not violated.

(ii) the number of hardware functional units, lifetimes of variables and additional multiplexer costs are minimized.

We assume that only single function modules, for example an add operator is processed by a hardware unit which implements an add function only, are used. Under this circumstance, if we can minimize the number of hardware functional units required to implement each type of operator then we can eventually minimize the total number of hardware units to implement the whole dataflow graph. Consequently, if we can assign all operators to all the available hardware units without logical, timing and resource conflicts[9], then we have found the cheapest feasible schedule.

4 Overview of NTUSS

In this paper we consider time-constrained scheduling with respect to minimizing the hardware costs only. Other considerations such as minimizing lifetimes of variables and minimizing multiplexer costs are not presented here. In our model, basically the problem of assigning off-critical path operators to any one of the timepoints within their ASAP and ALAP has been translated to :
'Squeezing' these operators into their respective resource utilization charts occupying the least height as possible without violating timing and data precedence constraints. We obtain the minimum height through a heuristic based algorithm which estimates the minimum number of hardware units required based on their mobilities (ALAP-ASAP times). It is not known to the author if it is possible to determine the absolute minimum number of hardware units based on their mobilities.

4.1 Resource Utilization Chart

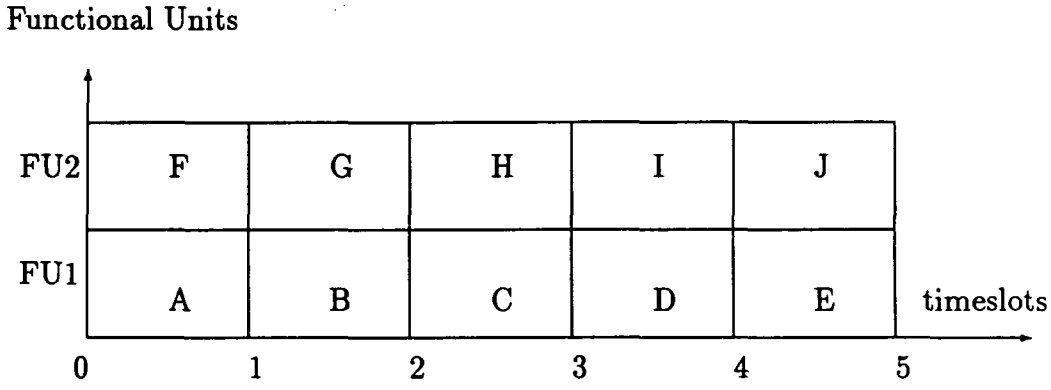

Figure 1: Resource Utilization Chart

A resource utilization chart which is equivalent to a Gantt Chart[10] shows in detail the usage of hardware functional units. Figure 1 shows such a chart. The vertical axis represents the type of hardware functional unit which process the operators that are placed in the chart. All hardware along this axis, for example FU1 and FU2 are identical. The height of the chart represents the total number of hardware units of this type required in the final implementation. The horizontal axis represents physical time which is also the schedule time. A timeslot occupied by an operator in this chart represents the starting time (schedule time) and the duration in which the operator in question is assigned to the stated hardware functional unit on the y-axis. Hence by filling operators in the chart, the algorithm does simultaneous scheduling and allocation. Each type of operator in the dataflow graph has its own resource utilization chart.

4.2 Algorithm to Determine the Minimum Number of Hardware Units

The minimum number of hardware units required to process a particular type of operators within their ASAP and ALAP times (mobility) is strongly dependent on their number and concentration along the timepoints. All critical path operators should also be taken into consideration. To estimate the minimum number, we concentrate our efforts on the timeslot which has the highest overlap of mobilities including that of critical path operators. We assume equal probability distribution of operators similar to the one adopted by Force-Directed Scheduling[5]. For example an operator with ASAP=0 and ALAP=2 will contribute a value of $\frac{1}{3}$(distribution density value) to each timeslot from 0 to 2. Critical path operators contribute a value of '1' to their respective timeslots. The timeslot with the greatest sum value (highest distribution density) is chosen. As shown in Figure 2, the sum may contain an integer portion and a fractional portion.

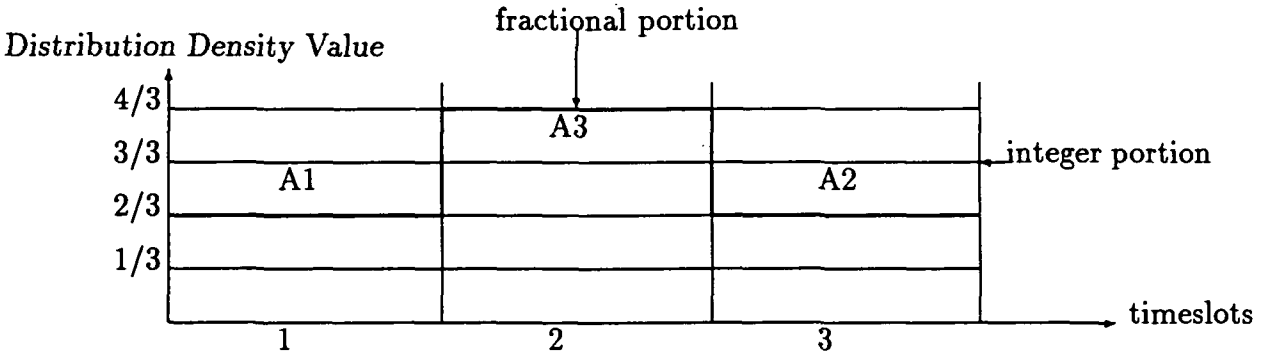

Figure 2: Distribution Density Chart

The highest distribution density value in this figure is $1\frac{1}{3}(1+\frac{1}{3})$. It can be said that $1\frac{1}{3}$,i.e., 2 machines can safely process all the operators whose distribution density values are shown in Figure 2. But we can provide a better estimate by the following method.

Using the integer portion as a reference, conceptually areas A1 and A2 represents the 'idle' time of the machine when it is operated from timeslot 1 to 3. Area A3 on the other hand, represents the amount of 'excess' work that one (reference) machine is incapable of handling at timeslot 2. If area A3 can be 'moved' to fill into areas A1 and A2 without any 'excess', then there is no need to employ an extra machine. This is the criteria used to determine whether the integer portion of the highest distribution density alone is sufficient to be taken as the minimum number or an extra '1' is required. The lower and upper limit of the x-axis of Figure 2 is obtained from step 3 and step 4 respectively in Figure 3. The algorithm to estimate the minimum number of hardware functional units for one type of operator is given in Figure 3.

1. Calculate Distribution Density value for each timeslot in the resource utilization chart.
2. Get timeslot with the highest distribution density value.
3. Get minimum ASAP time among all operators whose mobilities coincide with the timeslot obtained in step 2 (=mintime).
4. Get maximum ALAP time among all operators whose mobilities coincide with the timeslot obtained in step 2 (=maxtime).
5. maxvalue= greatest integer $\leq$ highest distribution density value.
   sum= 0;
   for (i= mintime; i $\leq$ maxtime; i++)
   {
   for (all operators whose mobilities lie within mintime and maxtime)
   /* this also includes operators in the critical path */
   {
   calculate sum(i)= distribution density value for timeslot(i) - maxvalue;
   sum= sum + sum(i);
   }
   }
   if (sum $\geq$ 1)
   minimum number of hardware units required= maxvalue + 1;
   else
   minimum number of hardware units required= maxvalue;

Figure 3: Algorithm to calculate the minimum number of hardware units.

The concepts presented so far can be integrated into a scheduling system whose algorithm is described in Figure 4.

1. Read in the records.
2. Create successor and predecessor list for each operator in the list.
3. Determine the maximum ALAP among operators in each group.
4. Calculate the minimum number of hardware units required for each group.
5. Create resource utilization chart for each group.
6. 'Enter' operators that are in the critical path into their respective resource utilization chart.
7. Sort the rest of the operators in order of decreasing processing times. If adjacent operators in the sorted list have equal processing times, then they are sorted again in order of ascending ALAP times.

8. Select the top (of the sorted list) operator's resource utilization chart.
9. Get the earliest available timeslot from the selected chart.
10. Go back to the sorted list and find the first operator which can legally be placed in the chosen timeslot. Steps 9 and 10 are repeated until an assignment is found.
11. Update mobilities of its predecessors and successors. Also mobilities of its predecessor's predecessors and successor's successors.
12. Update the resource utilization chart of those operators that has fallen into the critical path.
13. Go back to step 7 if there are some more unscheduled operators.

Figure 4 : The basic scheduling algorithm

5 Description of the Basic Algorithm

The basic technique used to 'fill' up the resource utilization chart is similar to the list-scheduling[8]. In list-scheduling used in Deterministic Scheduling Theory, tasks are ordered into a list based on some priority. When the processor becomes available, this list is scanned and the first unexecuted task which can be processed by this processor is then scheduled at that time and allocated to that processor. In our system, the list comprises of all the non-critical path operators. They are sorted in order of decreasing processing times. If adjacent operators in the list have equal processing times, they are then sorted in order of increasing ALAP times. The resource utilization chart of the top operator's type (in the main list) is chosen. The earliest available timeslot is selected (which means the earliest time the machine is free) and the main list of unscheduled operators is scanned for an operator that can be scheduled in that timeslot. If found, the operator is assigned to that timeslot and its predecessor's and successor's mobilities are adjusted to preserve data precedence relation. If not found, then the resource utilization chart is referred to again and the next earliest available timeslot is chosen. The main list is scanned again for a suitable operator. This process is repeated until a selection is made.

The resource utilization chart has to be 'filled' carefully in order to optimize hardware usage. For example in Figure 1, if only two hardware units are required, the sequence A, F, B, G, C, H, D, I, E and J (double-layered selection method) is found to be suitable for single-cycle operators. For two-cycle operators, the most suitable sequence is A, B, C, D, E, F, G, H, I and J (single-layered selection method). Since there could be more than one resource utilization chart to fill and only one chart is updated at any one time, there is a need for co-ordination among them. This is taken care of by ordering all the unassigned operators in the entire dataflow graph into one main list. The next resource utilization chart to be updated is chosen from this list. In this sense it is global in nature. Only when the chart has been chosen then only will the operators within the group of similar type be considered. Scheduling is complete when all the unassigned operators are 'entered' into their respective resource utilization charts.

The approximate worst case time complexity of the algorithm can be shown to be in the order of $4n^3 + 2n^2 - 2n$ where n is the total number of operators in the dataflow graph[3].

6 Experimental Results

Table 1 shows results of the benchmark example of fifth-order digital elliptical wave filter from Kung et al.[11]. In this example, 17 is the least cycle time that can be obtained from Critical Path Analysis at the expense of 3 adders and 3 multipliers. ALPS is able to obtain optimal results from their Linear Programming model. In order to reduce the hardware costs, cycle time is relaxed to 18 and then 21 cycles. NTUSS shows the least increase in the scheduling time taken.

SYSTEM	CYCLES	(+)	(*)	TIME	% CHANGE IN TIME
FDS[5] Xerox 1108 Lisp Machine	17	3	3	1 min	100 %
	18	3	2	3 min	300 %
	19	2	2	7 min	700 %
	21	2	1	13 min	1300 %
ALPS[7] Vax-11/8800	17	3	3	0.26 secs	100 %
	18	2	2	3.1 secs	1192 %
	21	2	1	34.5 secs	13269 %
[6] SUN4/280	17	3	3	0.067 secs	100 %
	18	2	2	0.101 sec	150 %
	21	2	1	0.783 sec	1169 %
NTUSS SUN3/60	17	3	3	1.25 secs	100 %
	18	2	2	1.30 secs	104 %
	21	2	1	1.42 secs	114 %

Table 1: Fifth Order Digital Elliptical Wave Filter[11]

7 Conclusions

A heuristic based scheduling algorithm is presented. Our algorithm differs from others. For example in MAHA[4], an operator is chosen based on its degree of freedom and then only a timeslot is chosen for this operator based on some other criteria. Our algorithm on the other hand, chooses a timeslot (which minimizes the idle time of the machine) first and then picks the most suitable operator from a sorted list. Some other heuristics like Force -Directed Scheduling[5] and the one described in [6], the most suitable (operator,c-step) pair is chosen based on some figure of merit. The decision making process is based on selecting the best figure among a group of computed values. Our algorithm minimizes numerical computation by making decisions based on direct usage of hardware resources in the resource utilization chart and storage device/multiplexer cost considerations (although storage device/ multiplexer costs considerations are described here). Using the scheme just described, NTUSS is capable of scheduling operators in almost linear time when their mobilities are increased.

There are also some limitations of the algorithm. For example it can handle static scheduling problems only. Currently, we are also not able to handle loops and allocation to pipelined datapaths. For the example presented we were able to obtain optimal results but like all heuristics, the same cannot be guaranteed for all cases.

We are able to incorporate storage device and multiplexer costs considerations into the model presented. Other real-world extensions like operator-chaining, multi-cycle and mutually-exclusive operations can also be handled.

Reference

1. M. C. McFarland, A. C. Parker and R. Camposano 'High Level Synthesis Of Digital Systems', Proc. of IEEE, vol 78, no.2 , February 1990.

2. B. M. Pangrle and D. D. Gajski 'Design Tools For Intelligent Silicon Compilation', IEEE Trans. Computer Aided Design pages 1098-1112, November 1987.

3. Gulam Mohamed 'A Heuristic-Based Scheduling Algorithm For High Level Synthesis Of Digital Systems', Master's Thesis, Nanyang Technological University (in preparation).

4. A. C. Parker, Jorge "T" Pizarro and Mitch Mlinar 'MAHA : A Program For Datapath Synthesis', Proc. 23rd Design Automation Conference, pages 461-466, July 1986.

5. P. G. Paulin and J. P. Knight 'Force-Directed Scheduling For Behavioral Synthesis Of Asics', IEEE Trans. Computer Aided Design, vol 8 pages 661-679, June 1989.

6. P. In-Cheol and K. Chong-Min 'Fast And Near Optimal Scheduling in Automatic Data Path Synthesis', Proc. 28th Design Automation Conference pages 680-685, June 1991.

7. Hwang C. T., Lee J. H. and Hsu Y. C. 'A Formal Approach To Scheduling Problem In High Level Synthesis', IEEE Trans. Computer Aided Design, pages 464-475, April 1991.

8. E. G. Coffman Jr. et al., **Computer And Job-Shop Scheduling Theory**, John Wiley & Sons, New York, 1976.

9. D. E. Thomas et al. , **Algorithmic And Register-Transfer Level Synthesis: The System Architect's Workbench**, Kluwer Academic Publishers, 1990.

10. K. G. Lockyer, **An Introduction To Critical Path Analysis**, The Pitman Press, Bath, 1969.

11. S. Y. Kung, H. J. Whitehouse and T. Kailath, **VLSI And Modern Signal Processing**, Englewood Cliffs, NJ, Prentice-Hall, pages 258-264, 1985.