List based scheduling

*Hardware and Software Codesign

Patrick NONKI
Electronics Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
patrick.nonki@stud.hshl.de

***Abstract*—In architectural synthesis, scheduling is a critical issue. A sequencing graph specifies simply the relationships between activities, however the scheduling of a sequencing graph specifies the exact start time of each activity. Because every set of actions associated by a sequency dependence (or by a chain of dependencies) cannot run concurrently, the start timings must fulfill the initial dependencies of the sequencing graph, which limit the level of parallelism of the operations.**
[1] Scheduling influences the performance of the eventual implementation because it determines its concurrency. Similarly, at any point in the schedule, the maximum number of concurrent operations of any given kind is a lower bound on the number of required hardware resources of that type. As a result, the selection of a schedule has an impact on the implementation area.
[1]

***Index Terms*—Scheduling, Latency, Resource, Jobs, Algorithm, Graph, Machine**

I. Introduction

When dedicated resources are used, unconstrained scheduling is used. As ogerations differ in kind or cost is minor when compared to steering logic, registers, wiring, and control, practical situations leading to dedicated resources are those.
When resource binding is done before scheduling, and resource conflicts are resolved by serializing activities that share the same resource, unconstrained scheduling is utilized. In this situation, the implementation's area cost is determined prior to and independently of the scheduling phase.
Unconstrained scheduling can eventually be used to derive latency bounds for constrained problems. Because the smallest latency of a schedule under some resource constraint is obviously at least as high as the latency estimated with unlimited resources, unconstrained scheduling can provide a lower bound on latency. [1]

II. ML-RCS and MR-LCS Algorithms

First, we'll look at the challenge of minimizing latency while dealing with resource restrictions, which is represented by vector a. This method is designed to deal with a variety of operation types as well as multiple-cycle execution delays.
[1] The selection step is used to categorize the list scheduling algorithms. Choosing among the processes is done using a priority list based on a heuristic urgency measure.
[1] Labeling the vertices with the weights of their longest paths to the sink and ranking them in decreasing order is a standard priority list. The most critical operations are prioritized. The algorithm is the same as Hu's algorithm when the operations have unit latency and there is just one resource type, and it offers an optimal solution for tree-structured sequencing graphs.
[1] Minimum timing limitations, in particular, can be addressed by postponing the selection of operations from the candidate set. The priority list is updated to reflect an unscheduled ooeration to a deadline tied to a maximum temporal limitation. By construction, the algorithms' schedules satisfy the required constraints. Needless to say, the heuristic nature of list scheduling may preclude the discovery of a viable solution.
[1] The goal of the list based scheduling is to attempt to solve the load balancing problem. One idea would be to try a greedy algorithm that consists of:

- Consider n jobs in arbitrary order
- Start with m empty machines
- Place the first job that on the machine that has the smallest amount of load (if 2 machines have the same amount of load place the job on either of them)
- Place the second job on the machine that has now the smallest amount of load ... and so on

But with this greedy algorithm we can see that all the jobs are not equally dispatched over the machine, then it becomes essential to find algorithms that give the same load for all the machines or approximately.

A. ML-RCS Algorithm (Minimum latency subject to resource bound)

Under latency constraints, list scheduling can also be used to reduce resource utilization. At first, it is believed that each type has only one resource, i.e., an is a vector with all elements set to 1. For this problem, the slack of an operation is the difference between the latest potential start time and the index of the schedule step under consideration, and the duck of an operation is used to rank the operations. The lesser the slack, the higher the list's urgency. The latency bound would be breached if operations with zero slack were not scheduled. Such activities may necessitate additional resources, such as upgrading a. The remaining activities will only be scheduled if they do not necessitate the use of additional resources.
[1] In order to understand this method the following algorithm is applied :

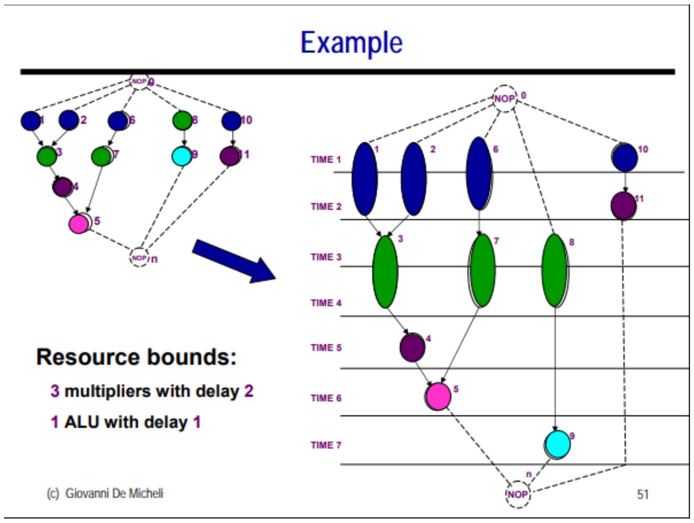

Fig. 1. Example of ML-RCS [1]

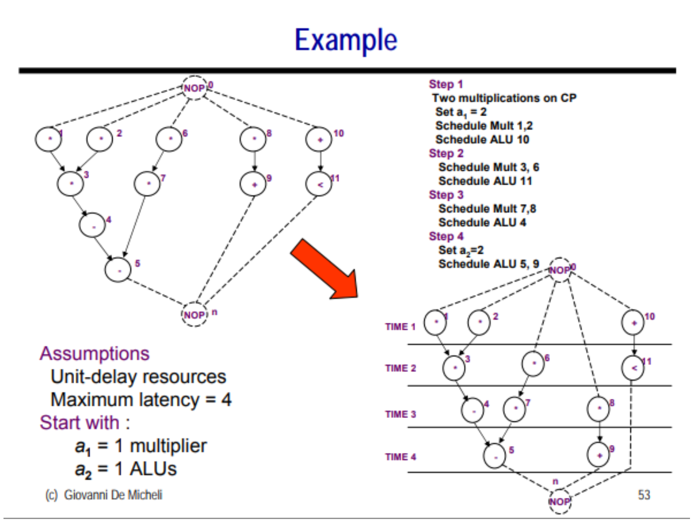

Fig. 2. Example of MR-LCS [1]

```
LIST_L( G(V, E), a) {
    l = 1;
    repeat {
        for each resource type k = 1, 2,     , nres {
            Determine ready operations Ul,k;
            Determine unfinished operations Tl,k;
            Select Sk     Ul,k vertices, s.t. |Sk| + |Tl,k|      ak;
            Schedule the Sk operations at step l;
            }
        l = l + 1;
    }
    until (vn is scheduled) ;
    return (t);
}
```

The example below shows the implementation of the ML-RCS Algorithm:

B. MR-LCS Algorithm (Minimum resource subject to latency bound)

In order to understand this method the following algorithm is applied :

```
LIST_R( G(V, E),    ) {
    a = 1;
    Compute the latest possible start times tL by ALAP ( G(V, E),    );
    if (t0 < 0)
        return (   );
    l = 1;
    repeat {
        for each resource type k = 1, 2,     , nres {
            Determine ready operations Ul,k;
            Compute the slacks { si = ti      l for all vi     Ulk};
            Schedule the candidate operations with zero slack and update a;
            Schedule the candidate operations not needing additionaL resources;
            }
        l = l + 1;
    }
    until (vn is scheduled) ;
    return (t, a);
}
```

The example below shows the implementation of the MR-LCS Algorithm:

III. Other similar algorithms

There is a lot of other list based scheduling algorithms that have been developed by the researchers over the years.Some of these methods have a common procedure; they all use the same phases [2]:

- The prioritizing phase: where each task is assigned a priority
- The selection phase: where each task is selected regarding its priority and the ready state
- The processor selection: where a processor is selected for running the chosen task.

A. CPOP Algorithm

The CPOP (Critical path on processor) algorithm is based on both upward and downward ranking techniques. It has 2 steps: The task prioritization and the task allocation phases.
On the task prioritization phase, they are prioritized based on their rank values(rank upgrading way + rank downgrading way) that are calculated based on the DAG (Direct acyclic node graph where all the tasks are represented by precedence) [2], and set in a table in a decreasing order.
On the allocation phase, they are selected based on the highest rank value and then assigned to the processor having the less processor cost.

B. HCPT Algorithm

This algorithm is divided in three steps:

- The level sorting phase: Where the DAG is traversed from up to down to sort and group tasks on each level that are independent.
- The task prioritization phase: Where tasks are prioritized based on the ranking values as shown in the CPOP.
- The processor selection phase: Where the processor is selected based on the EFT (Earliest finish time) value of each task on a given processor [3].

C. HPS Algorithm

The HPS (High Performance task Scheduling) algorithm uses the same steps as the one above. As a major change, we can note that:

- In the task prioritization phase, priorities are assigned based on the Down link cost (DLC) that is the maximum communication cost from a task to his predecessors, the up link cost (ULC) that is the maximum communication cost from a task to his successors, and the link cost (LC) that is the sum of DLC and ULC. The highest priority is then given to the task with the highest LC [2].

IV. Conclusion

Overall, list scheduling methods have been widely employed in synthesis systems due to the algorithm's low computational complexity, which allows it to work with enormous graphs. For those (limited) situations where the ideal solutions are known, solutions have been proven to be quite similar in latency to the optimum ones.

Acknowledgment

This work has been made possible at the Hochschule Hamm-Lippstadt under the course Hardware Software Codesign and supervised by Pr.Dr Achim Rettberg. Thanks also to Abdul-Azeez Olanlokun for the review that has contributed to make this paper better. [1]

References

[1] Janan Paremajalu. G. De Micheli Synthesis And Optimization Of Digital Circuits (Text Recognized Using OCR) [v. 1.03 20 4 2005].
[2] Hamid Arabnejad. List Based Task Scheduling Algorithms on Heterogeneous Systems - An overview. page 10.
[3] C Subramanian, N Rajkumar, and S Karthikeyan. LIST BASED SCHEDULING ALGORITHM FOR HETEROGENEOUS SYSYTEM. *2015*, 10:6, 2015.