Scheduling Algorithms

Joyen Benitto

Scheduling

Scheduling is a very important problem in architectural synthesis. While a sequencing graph does a good job at capturing the order of operation and dependencies among these operations but scheduling is responsible for annotating the start time of these operation also remember that the scheduling has to be aware that concurrency is possible but only when the ops are not dependent and can be parallelized so the behavior is preserved.

Scheduling determines the concurrency of the resulting implementation and therefore it affects its performance. By the same token at any given point of time the total number of concurrent operations of any given type at any step of the schedule is a lower bound on the number of required hardware resources of that type. So the choice of schedule also affects the area of the implementation.

Continuing on the discussion from above, the scheduling graph with resource bounds are done in order to meet area constraints. when the number of resources are bounded even if there are parallel operation we will have to serialize the extra operation as no resources are available for computation. A bare minimal case is only one operation available in one go in such a case execution is linear.

Area/latency trade-off points can be derived as solutions to constrained scheduling problems for desired values of the cycle-time. The area evaluation is just the weighted sum of all the resource in resource dominated circuit whereas for other circuits we need to consider the the corresponding to steering logic, registers, wiring and control area.

A model for the scheduling problems

In the following section we will be using a non-hierarchical sequencing graph model. Before we go ahead a sequence graph is a polar directed acyclic graph $G_s(V, E)$ where the vertex set $V = \{v_i; i = 0, 1, 2, ..n\}$ denotes the set of operations and is in one to one correspondence with $E = \{(v_i, v_j); i, j = 0, 1, 2...n\}$ represents dependencies. n denotes the total number of operations and $n = n_{\text{ops}} + 1$ and we will denote the source verte by v_0 and the sink by v_n; both are NOP. Let $D = \{d_i; i = 0, 1, 2, 3, 4..n\}$ be the set of execution delays and for simplicity and for understanding purposes we will for the time-being consider all the delays as known and keep the delays independent of the data. The execution delays of the source and the sink is $d_0 = d_n = 0$. The time of execution is denoted by $T= {t_i; i= 0, 1, 2, 3.. n}, the *start time* for the operations. λ denotes the latency of the schedule; it is the cycles taken to execute the entire schedule form the source to the sink. we also assume the first operation starts at cycle 1 i.e in the first cycle.

$$\lambda = t_n - t_0 \qquad (t_0 = 1)$$

The sequence graph requires that the start time of an operation is at leasst as large s teh start time of each of its direct predecessor plus its execution delay, i.e the following relation holds

$$t_i \geq tj + dj, \qquad \forall i, j : (v_j, v_i) \in E$$

ASAP scheduling model

The ASAP algorithm is useful in cases of unconstrained scheduling, in the sense that you are not bound by resources just by the order of execution and parallelism dictated by data dependencies. The algorithm is uselful when we have a dedicated resource for every operation, the following model is good when you have a algorithm's who's steering logic is more costly than the dedicated resource itself.