As-Soon-As-Possible (ASAP) Scheduling

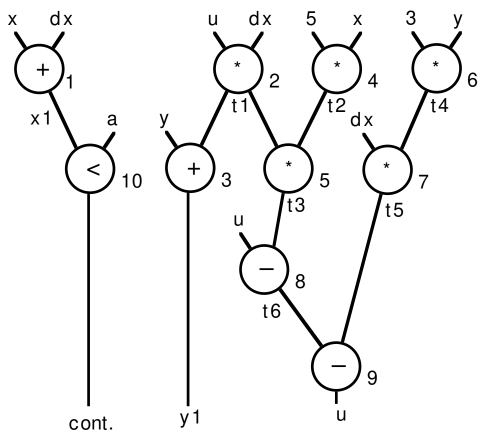
for each operation o_i

if o_i has no immediate predecessors

assign o_i to cstep 1

else

assign o_i to (maximum cstep of any of o_i's predecessors) + 1



As-Late-As-Possible (ALAP) Scheduling

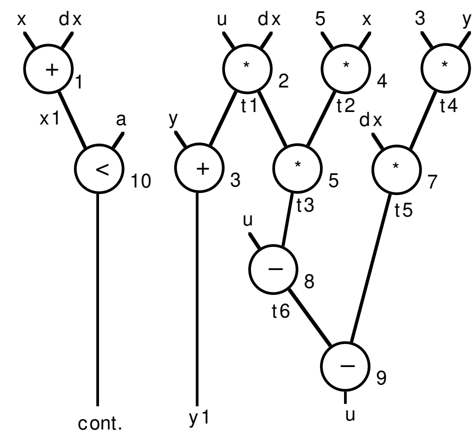
for each operation o_i

if o_i has no immediate successors

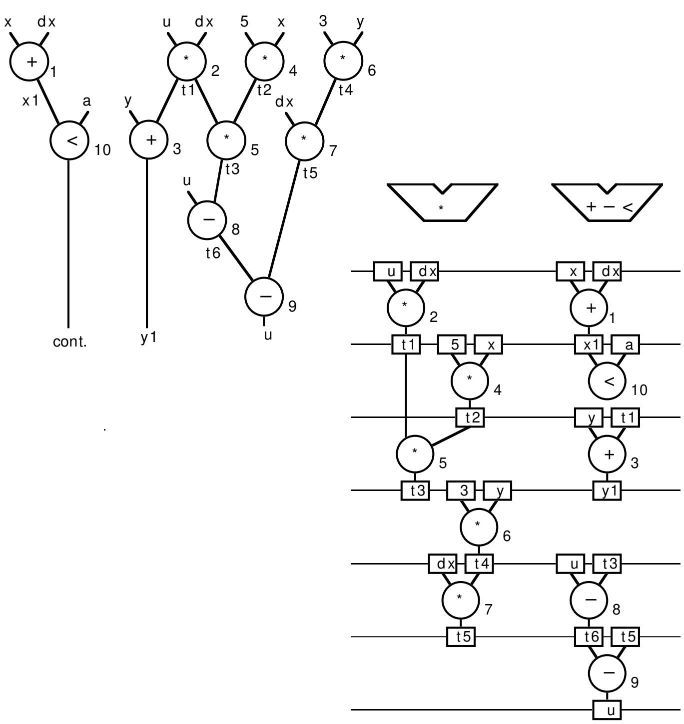
assign o_i to cstep d

else

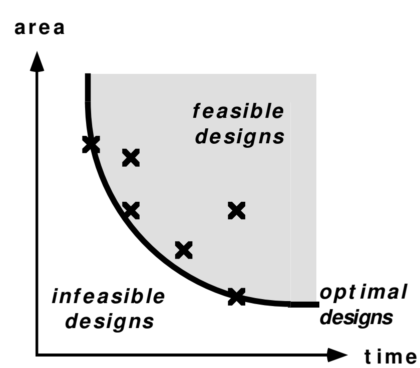
assign o_i to (minimum cstep of any of o_i's successors) – 1



Another Possible Schedule (One Multiplier, One ALU (+,–,<))



The Design Space



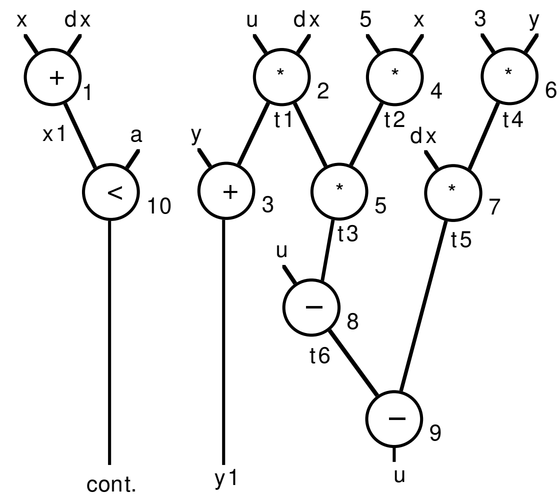
- For optimal designs, there is a tradeoff between:
  - *time* (schedule length), and
  - *area* (ideally total area, but usually simplified to functional unit area)
- We'd prefer to find *optimal* designs, but a heuristic (such as ASAP scheduling) only guarantees *feasible* designs

Three Scheduling Problems

- ***Scheduling*** is the problem of determining the ***control step***, or state, in which each operation will execute
- The scheduling problem is usually specified in one of three ways, depending on the desired goal:
  - ***Time-Constrained Scheduling*** (TCS) — for a fixed schedule length, minimize the number of resources (functional units)
  - ***Resource -Constrained Scheduling*** (RCS) — for a fixed number of resources (functional units), minimize the schedule length
  - ***Time- and Resource-Constrained Scheduling*** (TRCS) — for a fixed schedule length, and a fixed number of resources, find a feasible (or optimal) schedule

Example of Resource-Constrained Scheduling

- Schedule this DFG, assuming there are only 2 multipliers and 2 ALUs (+,–,<) available



- How could the ASAP algorithm be modified to solve this problem?

List Scheduling (To Solve the RCS Problem)

evaluate the priority of each operation

current-cstep = 1

while there are unscheduled operations

  current-cstep = current-cstep + 1

  place data-ready operations into the ready list

  sort the ready list in order of priority

  while there are data-ready operations in the ready list that meet the resource constraints

    choose the highest priority data-ready operation o_i from the ready list

    assign o_i to current-cstep

Notes on List Scheduling

- Solves the RCS problem
- Basic operation differs from ASAP:
  - ASAP — processes <u>operations</u> in a fixed order
  - List Scheduling — processes <u>csteps</u> in a fixed order
    - Fill one cstep, then go on to the next
- Uses a *ready list* to keep track of *data-ready operations* — those unscheduled operations that can be scheduled into the current cstep without violating:

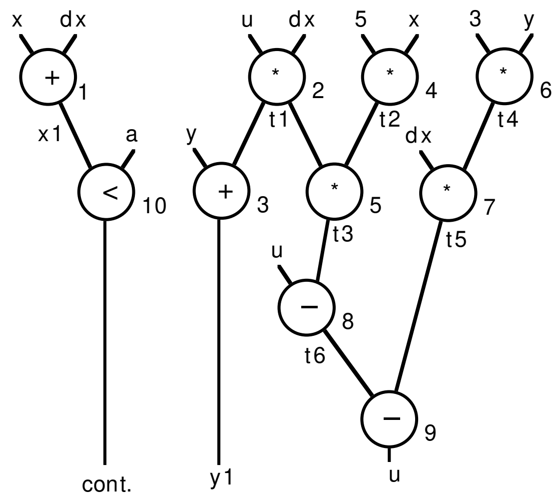
  precedence constraints (data dependencies)

  resource constraints

  - Pick operations from this ready list, and schedule them into the current cstep until it is full (i.e., other operations would violate the resource constraints)

List Scheduling Example

- Use list scheduling to schedule this DFG with a resource constraint of 2 multipliers, and 2 ALUs (+,–, <)



Notes on List Scheduling (cont.)

- As each cstep is processed, the data-ready operations are sorted according to *priority*
  - Data-ready operations are then removed from the ready list and scheduled into the current cstep based on their priority
- Common priority functions, giving increased priority to operations with:
  - Lower *mobility* — length of operation's *schedule interval* (ALAP – ALAP + 1)
  - Longer path to end of graph
  - Greater number of immediate successors
- We will use:
  - Primary priority function: highest priority to operations with lower mobility
  - Secondary priority function: highest priority to operations parsed earlier

Other Scheduling Issues

- TCS scheduling
  - Force-directed scheduling
- Optimal scheduling
  - ILP formulations
- Control step length
  - Chaining
  - Multi-cycling
  - Pipelining
- Design space exploration
  - Control step length
  - Module set
  - Need better area estimation