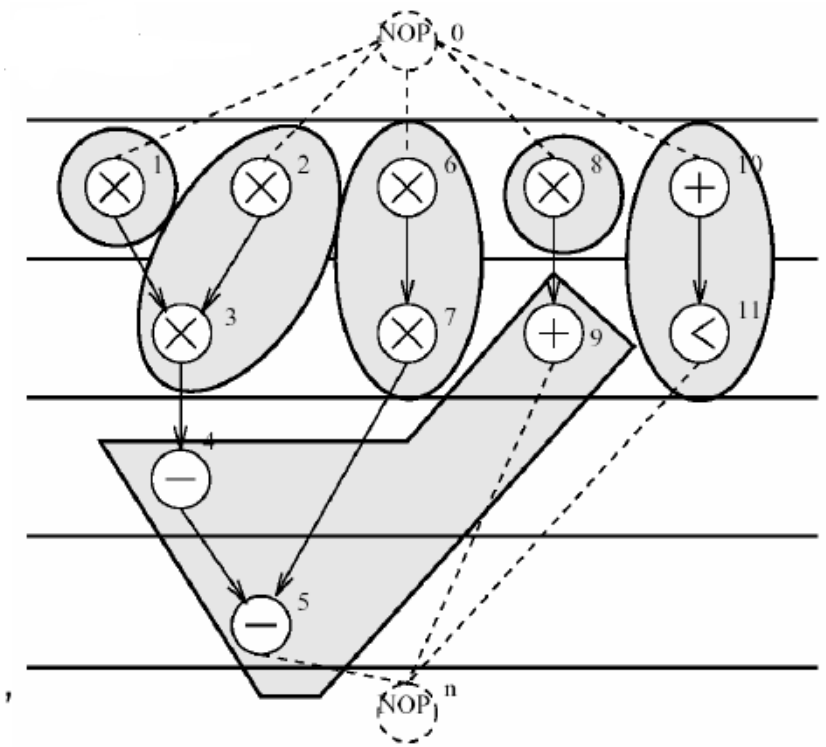
EN2911X: Reconfigurable Computing
Lecture 12: Design Flow: Resource Binding (4)

Prof. Sherief Reda
Division of Engineering, Brown University
http://scale.engin.brown.edu
Fall ’09

BROWN

Resource sharing and binding

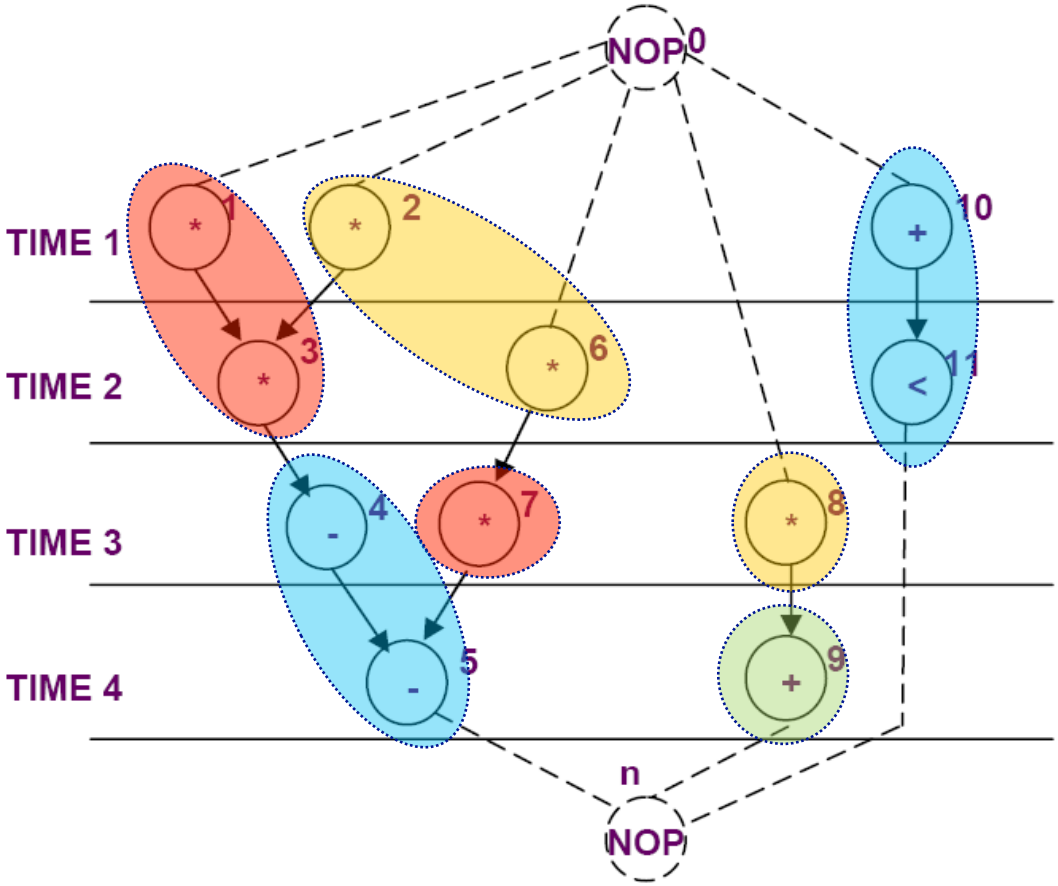
- Bind a resource to two operations as long as they do not execute concurrently



How many instances of the multiplier and the ALU do we need now?

BROWN

Can we do better? Can we get the same latency with less resources

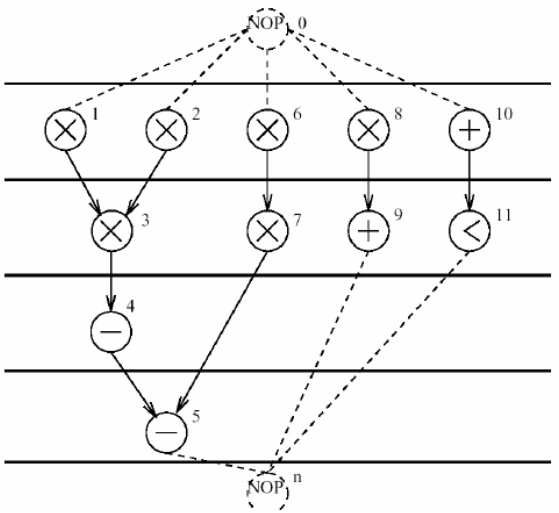


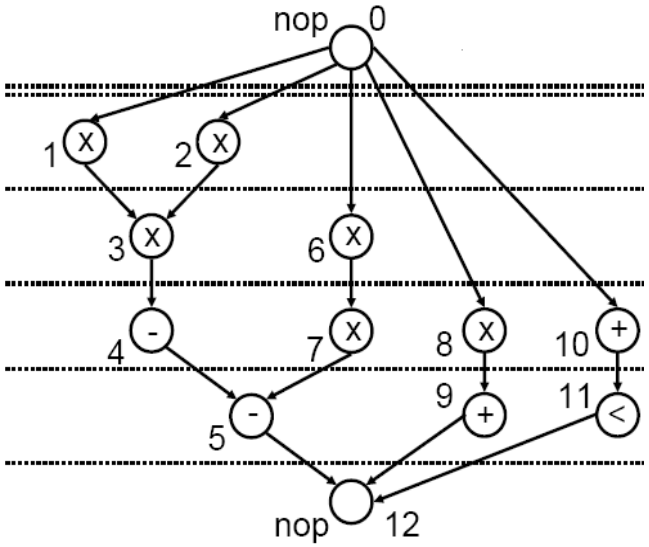
Resources sharing the same instance are colored with the same color.
How many instances are now needed? How can we find the solution?

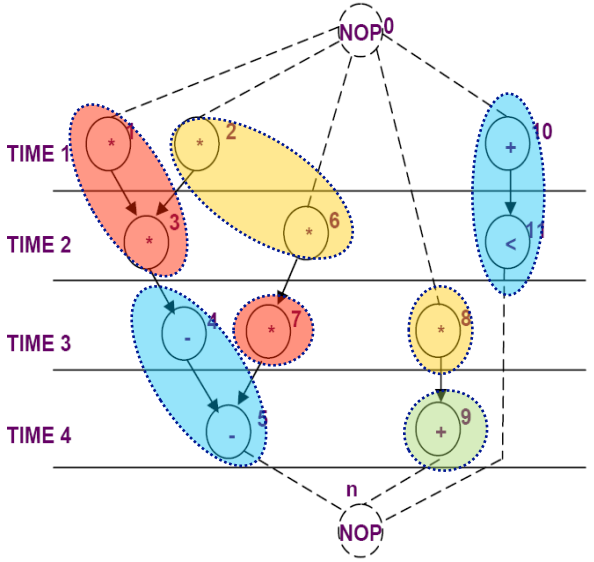
BROWN

Finding the minimal number of resources for a given latency (T) using list scheduling

- Initialize all resource instances to 1.
- for t = 1 to T: For each resource type:
  - Calculate the *slack* (ALAP time – t) of each *candidate* operation
  - Schedule candidate operations with zero slack and update the number of resource instances used if needed
  - Schedule any candidate operations requiring no new resource instances



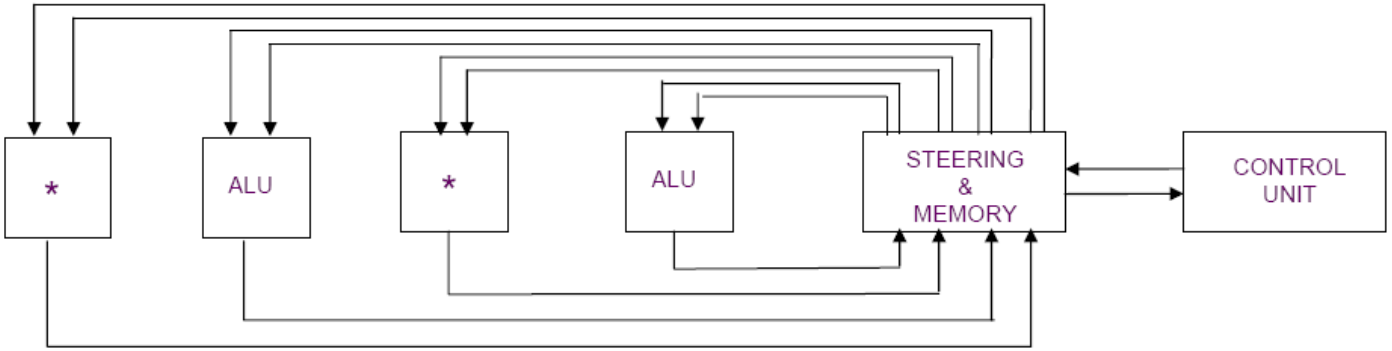


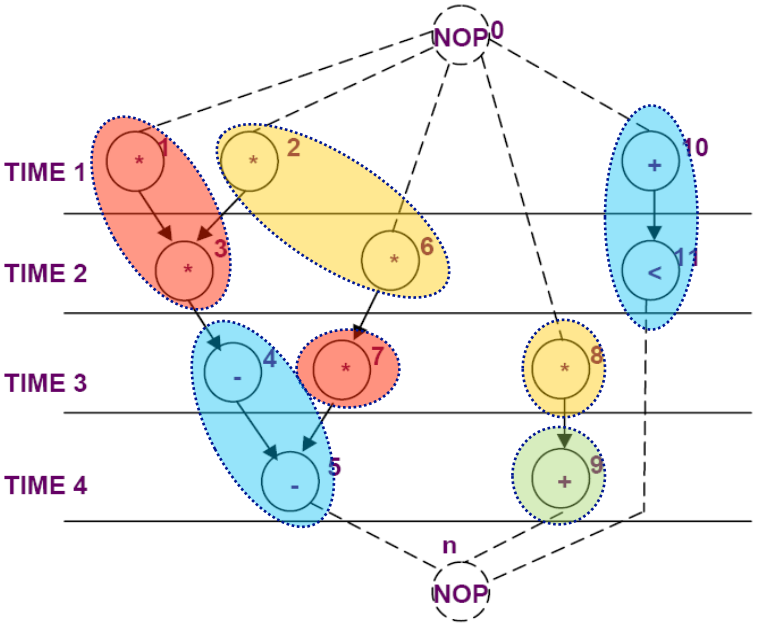


What is the intuition behind this heuristic?

BROWN

Scheduling and sharing necessitates a control unit that orchestrates the sequencing of operations

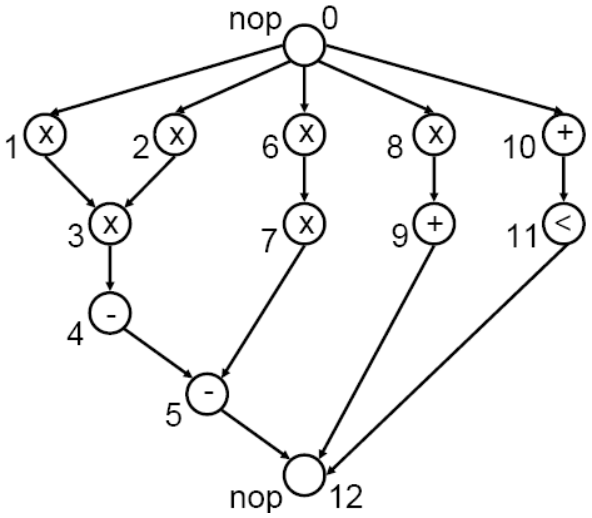


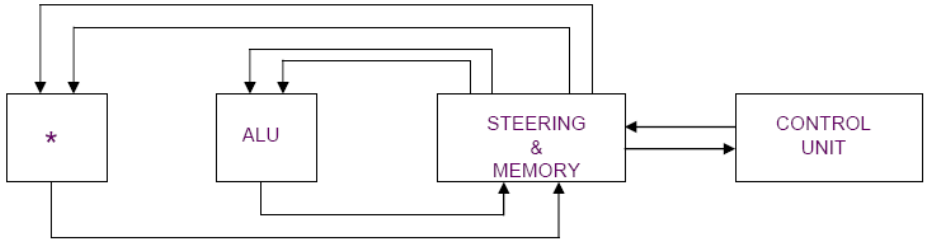


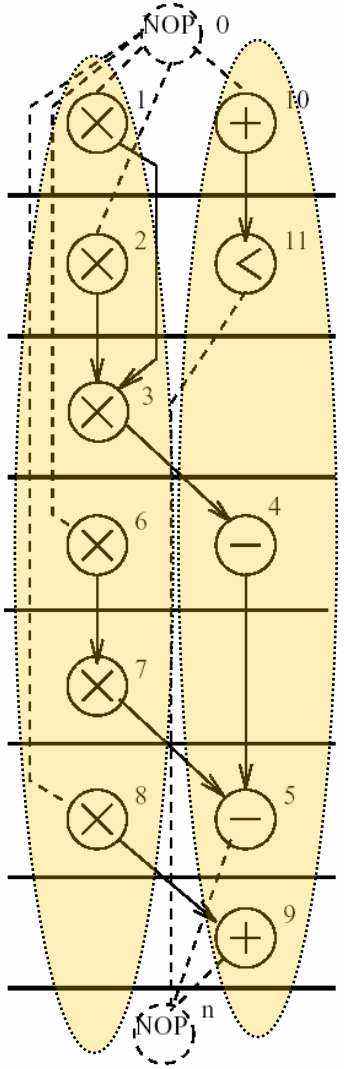
BROWN

Scheduling under resource constraint

Assume we just one instance of a multiplier and one instance of an ALU (+, - and ==), how can we schedule all operations? What is the latency?



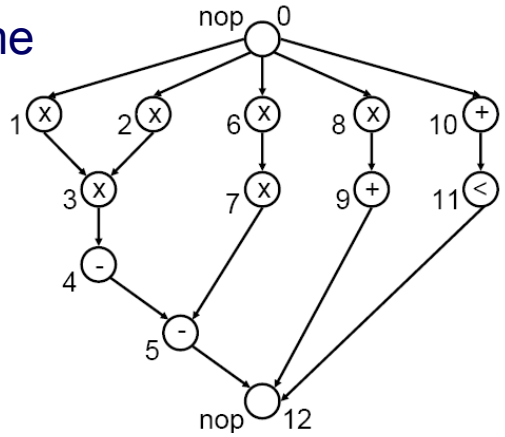


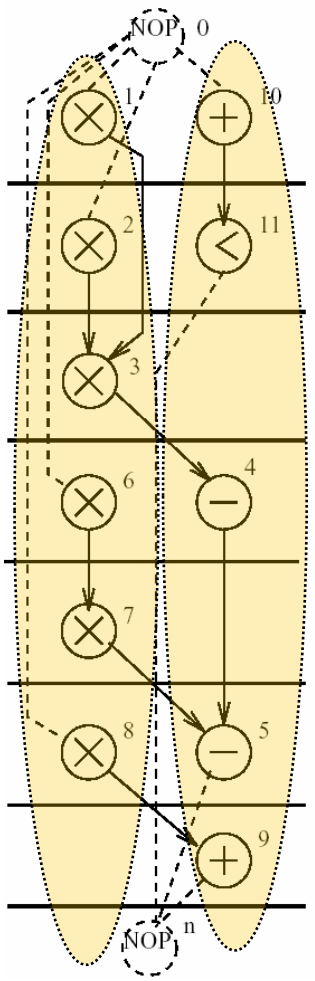


BROWN

Finding the minimal latency for a given resource constraint (C) using list scheduling

- Label all operations by the length of their longest path to the sink and rank them in decreasing order
- Repeat
  - For each resource type
    - Determine the candidate operations that U can be scheduled
    - Select a subset of U by priority such that the resource constraint usage (C) is not exceeded
  - Increment time

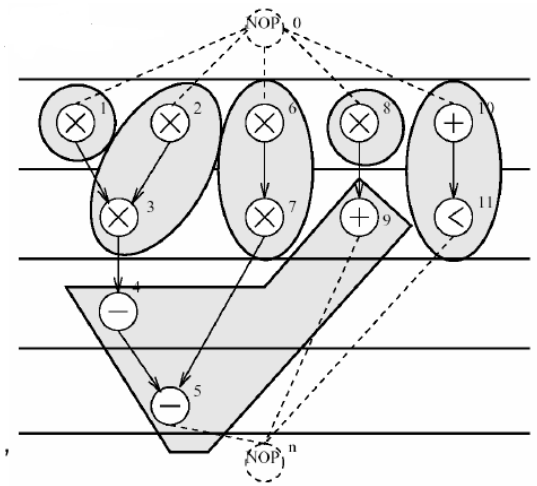


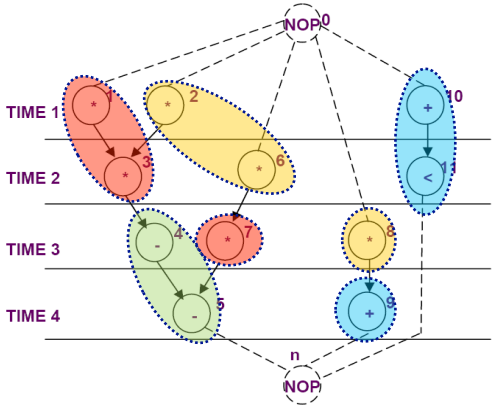


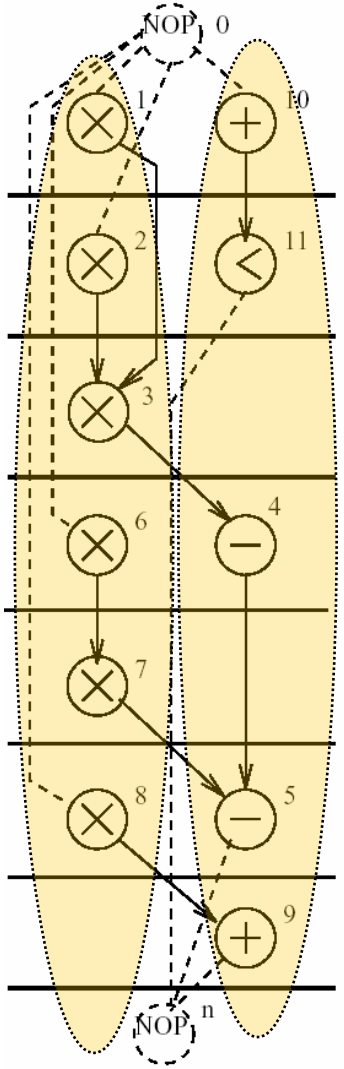
What is the intuition behind this heuristic?

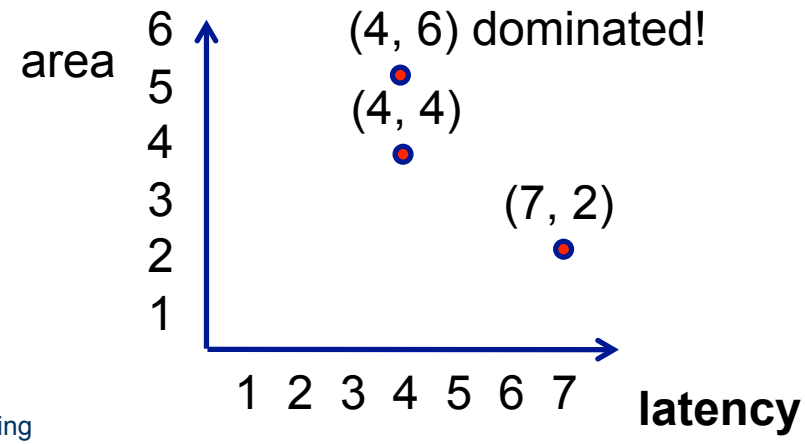
BROWN

There is an inherent tradeoff between area and latency









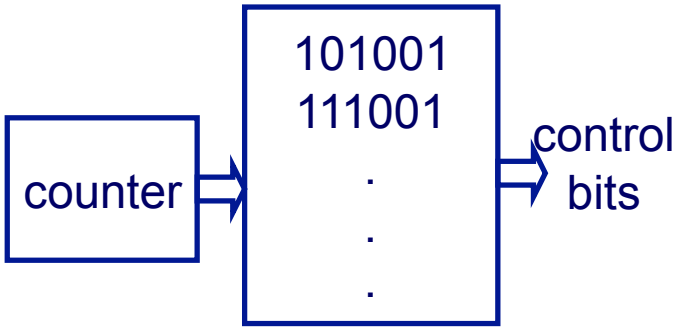
BROWN

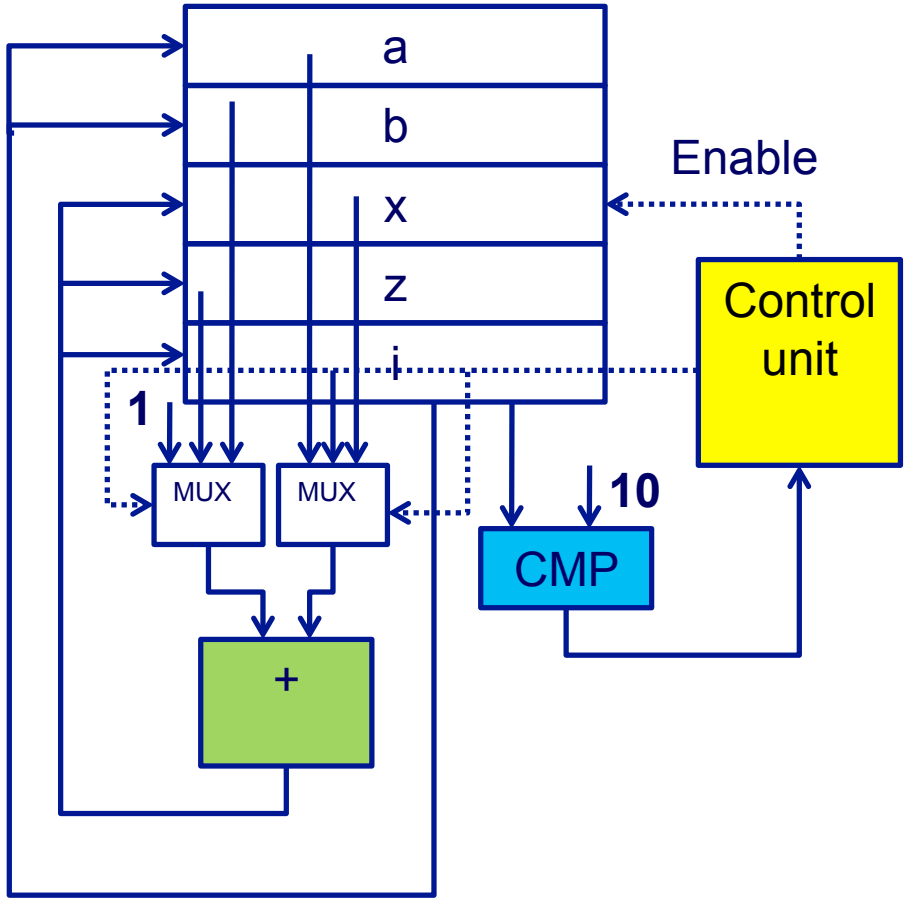
Control unit example

```
for(i=0; i<10; i=i+1)
begin
  x = a[i] + b[i];
  z = z + x;
end
```

How many control signals are produced from the control unit?

How can we design the control unit?





BROWN