Retiming

Transform register placement for speed
Placement in flow depends on accuracy vs. ease of restructuring.
Retiming Transformation

Move Gate:

Forward in time (+1)

Backward in time (-1)
Nodes are gates. Assign values to nodes to retimeme them (move registers)
Legality Restrictions

What happens if an edge has no registers, and you then try to remove one?

If A is \(+\Delta\), E must be at least \(+\Delta\)

If E is \(-\Delta\), A must be at least \(-\Delta\)

If C is \(+\Delta\), F must be at least \((\Delta + 1)\)

If F is \(-\Delta\), C must be at least \(-(\Delta + 1)\)
Legal Retiming Graph

No edge can have # of flip flops on it.
Cycle Times

To reach a cycle time of say 2ns (all gates 1ns), paths > 2ns must have at least one register.
Retiming Graph for 2ns Clock
Retiming Graph for 1ns Clock
I/O Interfaces

For 1ns clock, we “stole” registers from the I/O – fair?

Environment
Delay = clock period
Basic Retiming Limitations

For a latency-tolerant design (where you can add as many registers at the I/Os as needed) what is the limit on performance via basic retiming?

\[ \sum \text{gate delays on cycle} \]

\[ \sum \text{register DFFs on cycle} \]
Approach 1: Resynthesis & Retiming

Restructure circuit to remove logic from the cycles.
Approach 2: C-slow Retiming

Replace every register in the design by C registers
  + More registers on each cycle
  - Changes behavior of circuit