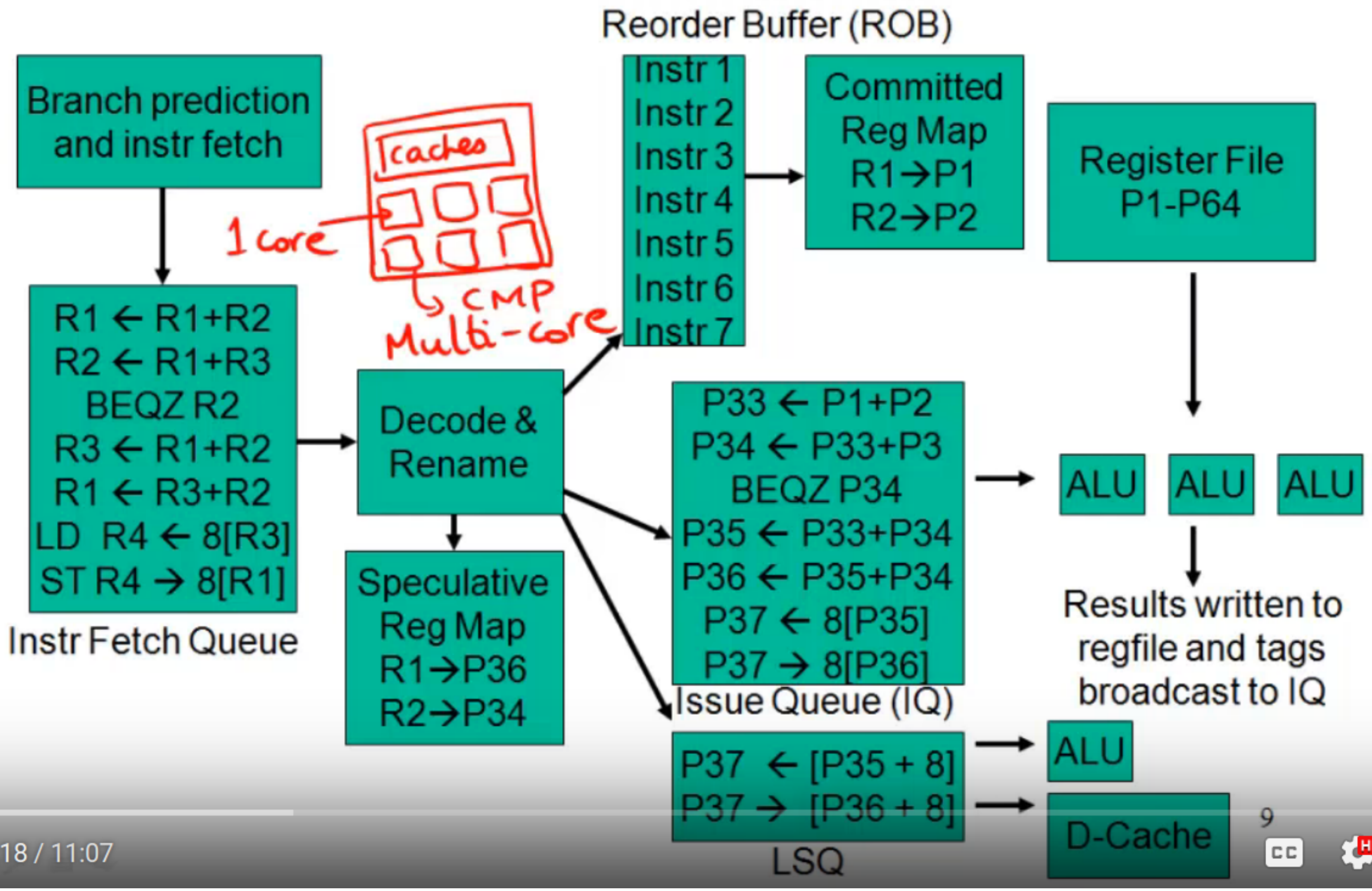
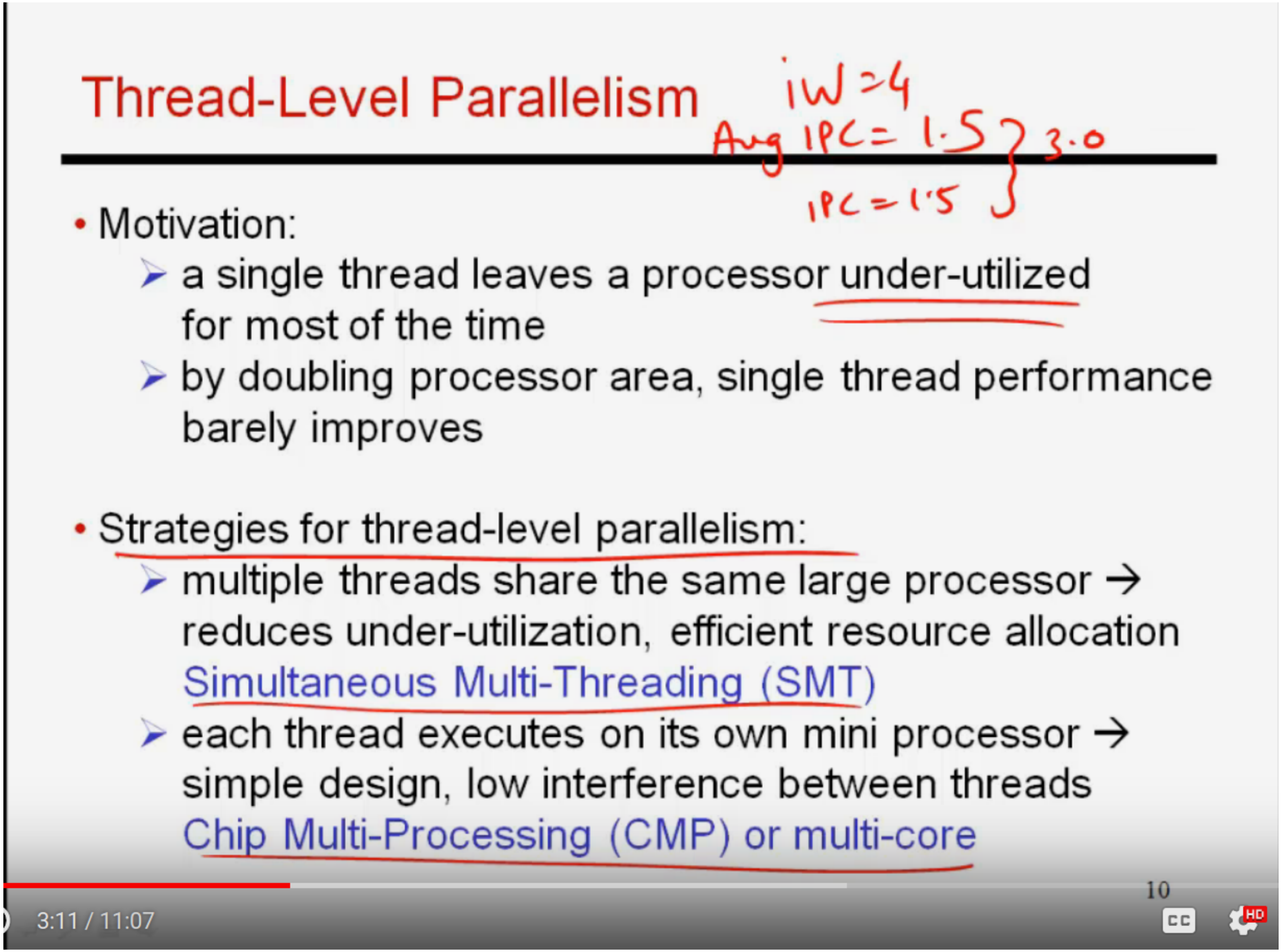
# The Alpha 21264 Out-of-Order Implementation





## Thread-Level Parallelism

iW = 4

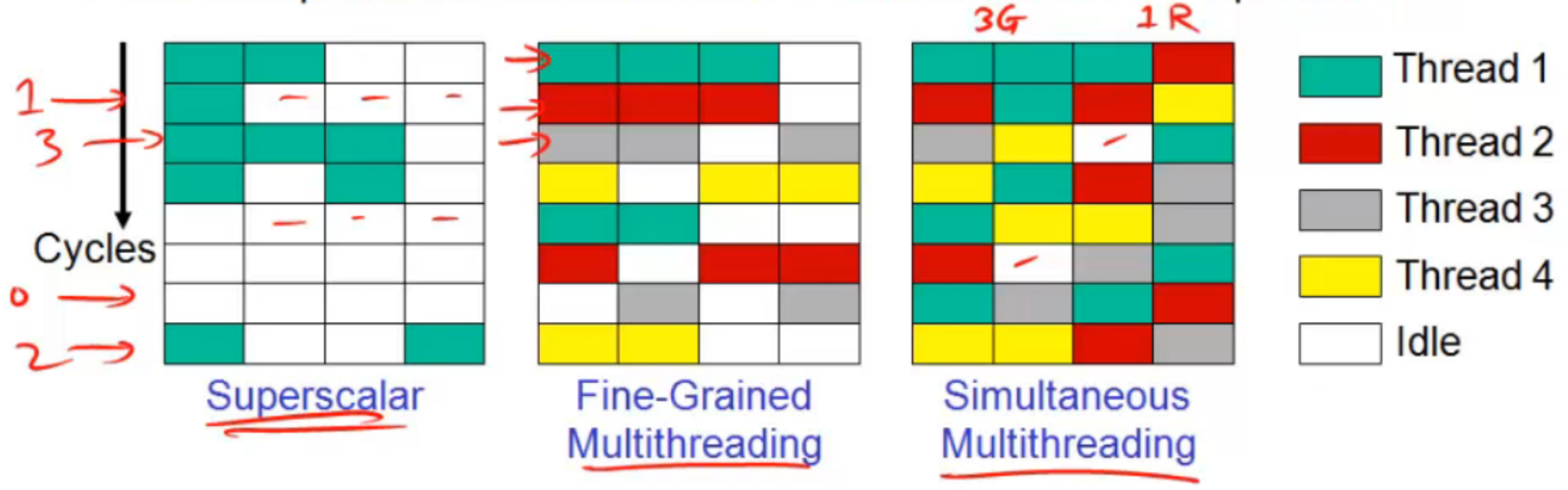
$$\left.\begin{array}{l}\text{Avg IPC} = 1.5 \\ \text{IPC} = 1.5\end{array}\right\} 3.0$$

- Motivation:
  - a single thread leaves a processor under-utilized for most of the time
  - by doubling processor area, single thread performance barely improves

- Strategies for thread-level parallelism:
  - multiple threads share the same large processor → reduces under-utilization, efficient resource allocation
    Simultaneous Multi-Threading (SMT)
  - each thread executes on its own mini processor → simple design, low interference between threads
    Chip Multi-Processing (CMP) or multi-core

## How are Resources Shared?

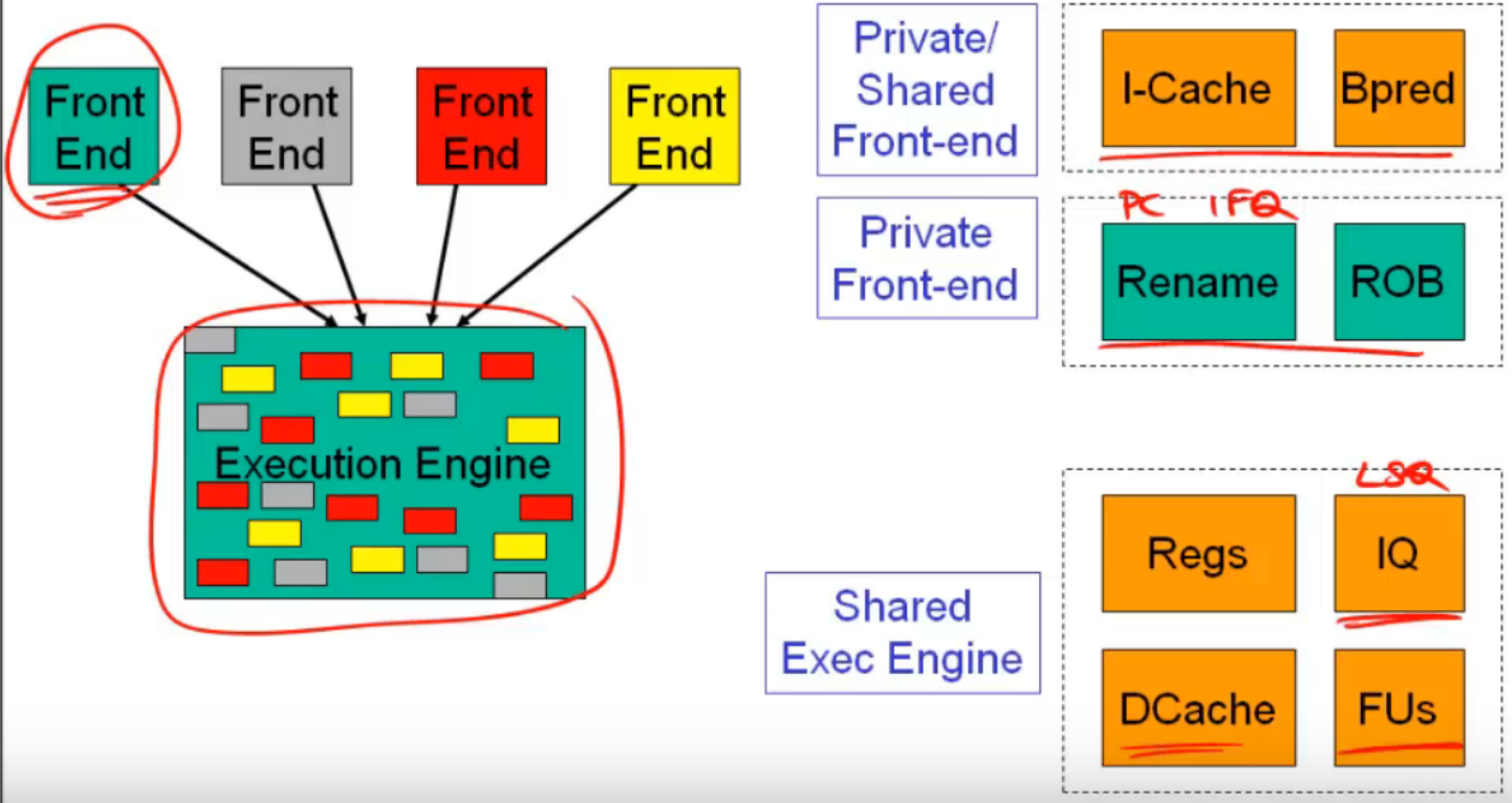
Each box represents an issue slot for a functional unit. Peak thruput is 4 IPC.



- Superscalar processor has high under-utilization – not enough work every cycle, especially when there is a cache miss
- Fine-grained multithreading can only issue instructions from a single thread in a cycle – can not find max work every cycle, but cache misses can be tolerated
- Simultaneous multithreading can issue instructions from any thread every cycle – has the highest probability of finding work for every issue slot

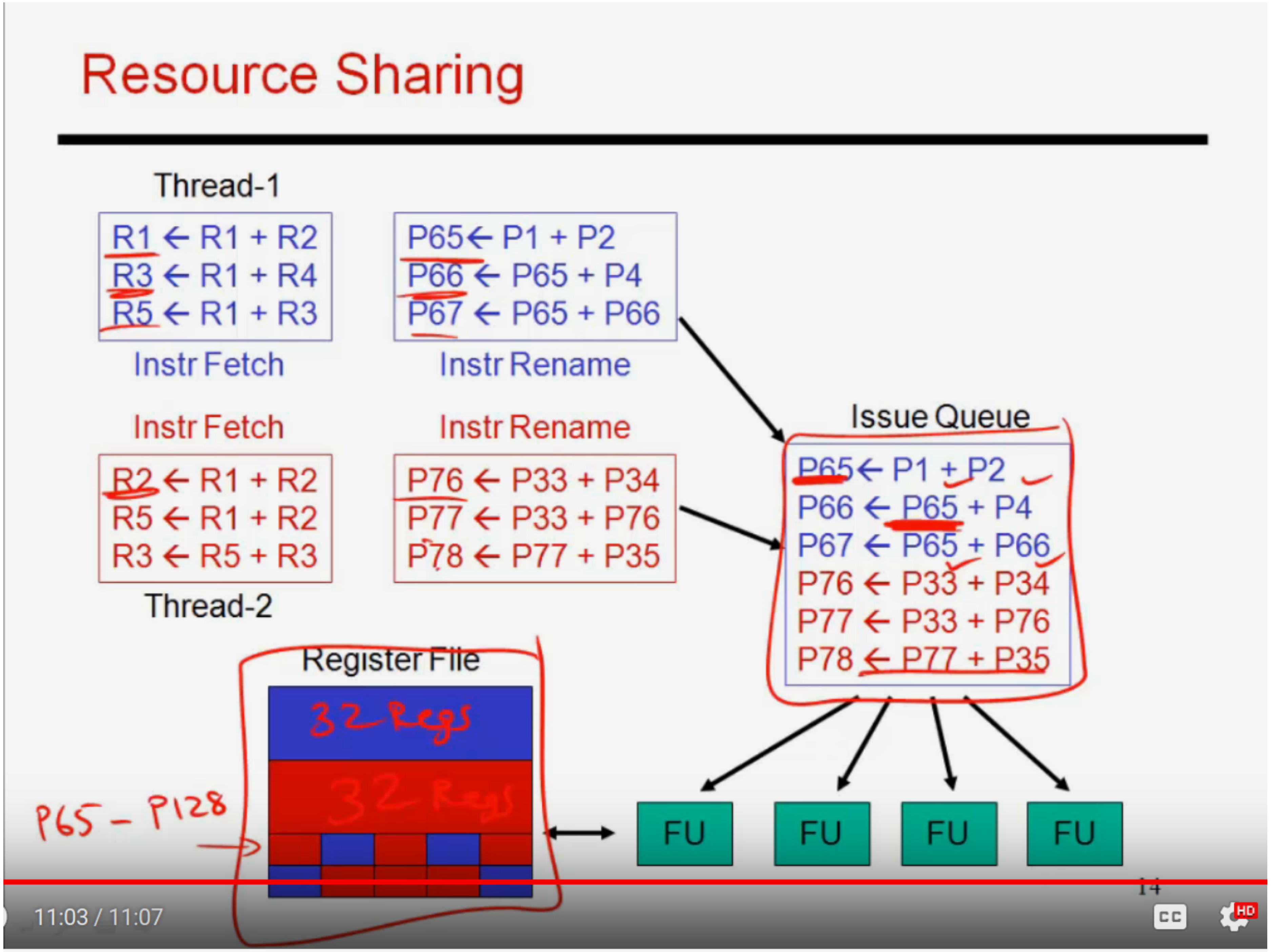
4:49 / 11:07

# Pipeline Structure



9:13 / 11:07

## Resource Sharing



Thread-1

| Instr Fetch | Instr Rename |
|---|---|
| R1 ← R1 + R2 | P65 ← P1 + P2 |
| R3 ← R1 + R4 | P66 ← P65 + P4 |
| R5 ← R1 + R3 | P67 ← P65 + P66 |

Thread-2

| Instr Fetch | Instr Rename |
|---|---|
| R2 ← R1 + R2 | P76 ← P33 + P34 |
| R5 ← R1 + R2 | P77 ← P33 + P76 |
| R3 ← R5 + R3 | P78 ← P77 + P35 |

Issue Queue

P65 ← P1 + P2
P66 ← P65 + P4
P67 ← P65 + P66
P76 ← P33 + P34
P77 ← P33 + P76
P78 ← P77 + P35