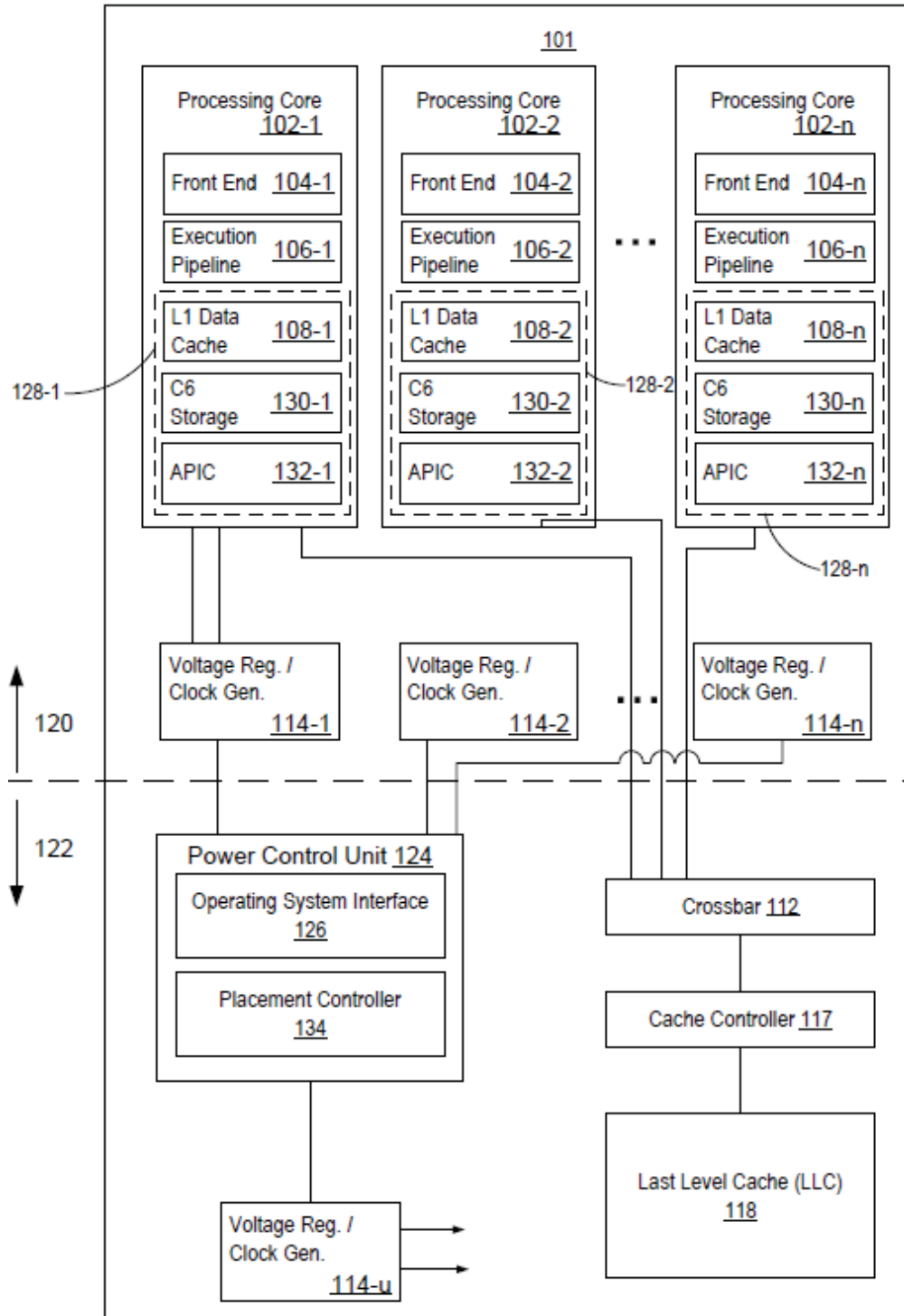




100

Module
99

FIG. 1

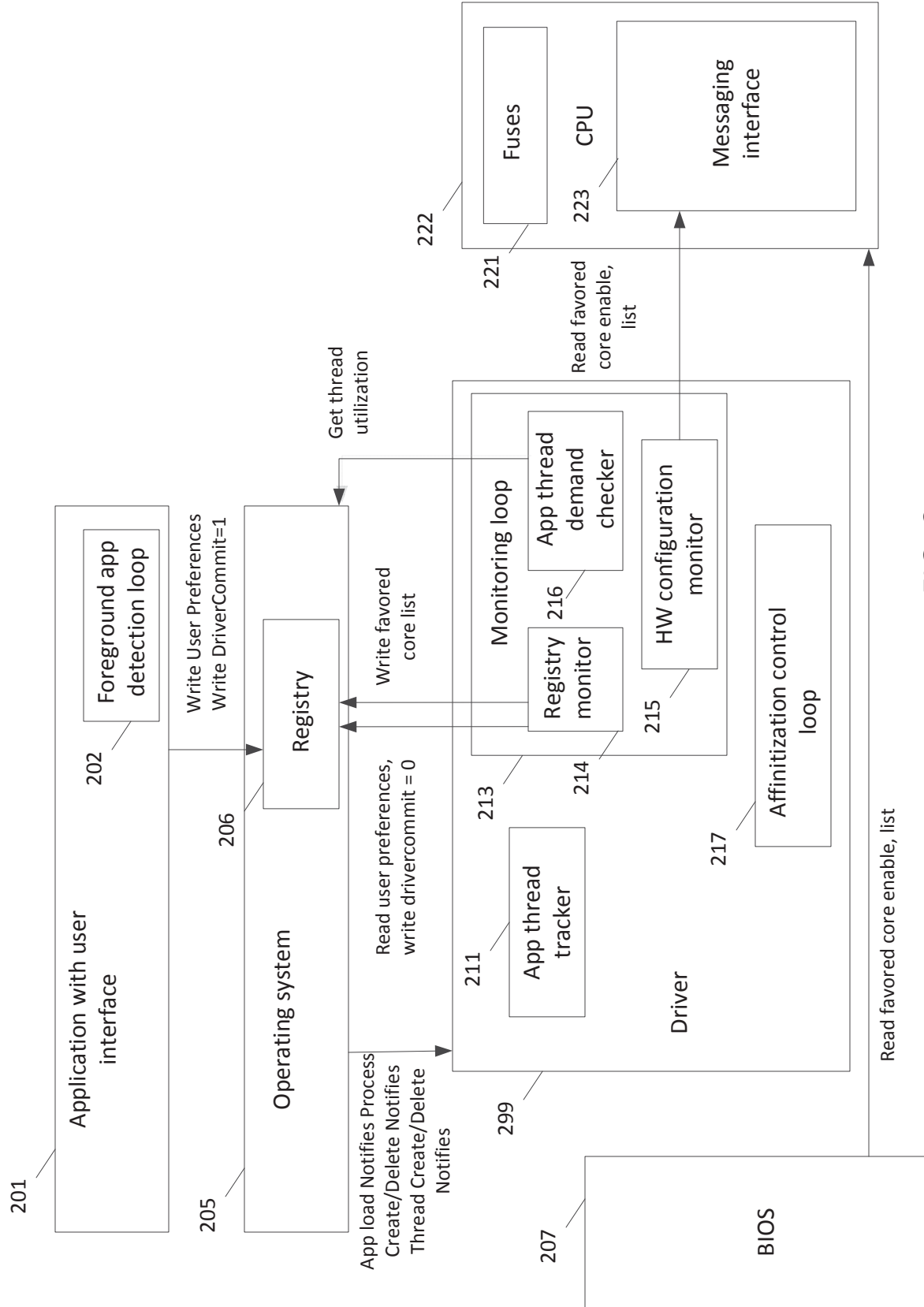


FIG. 2

Driver process – application thread tracker

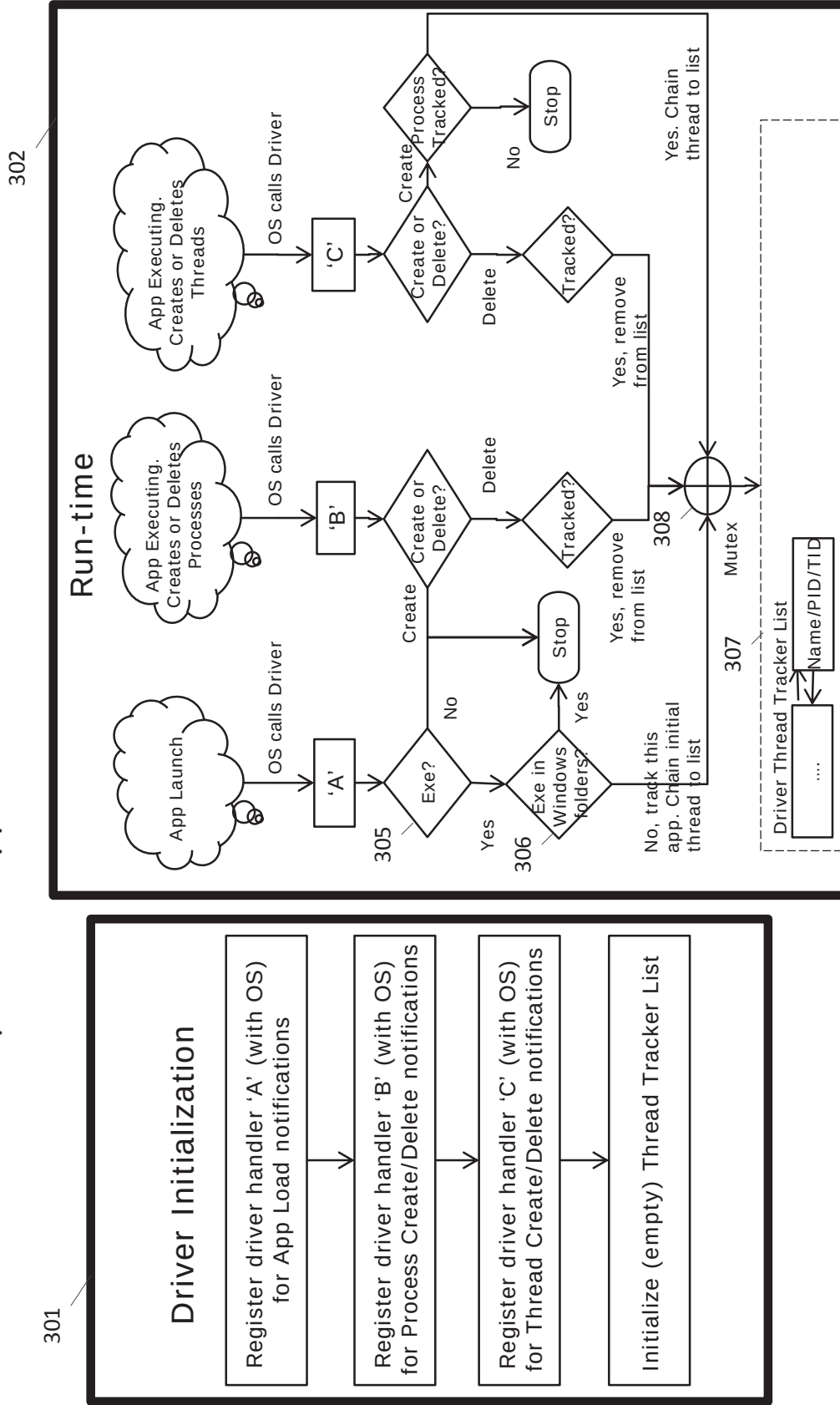


FIG. 3A

Driver process – registry/hw config monitor loop

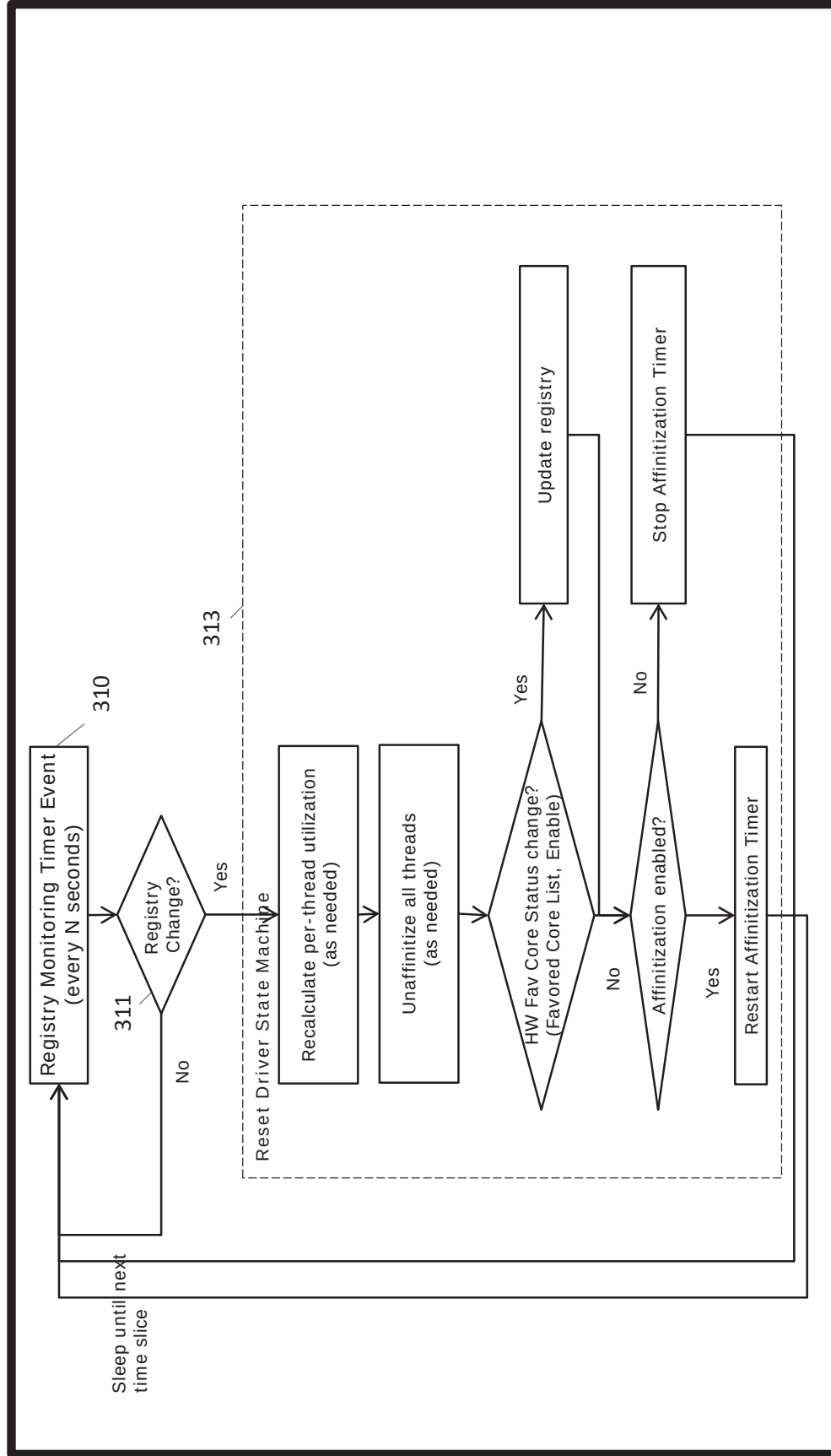


FIG. 3B

Driver process – affinitization control loop

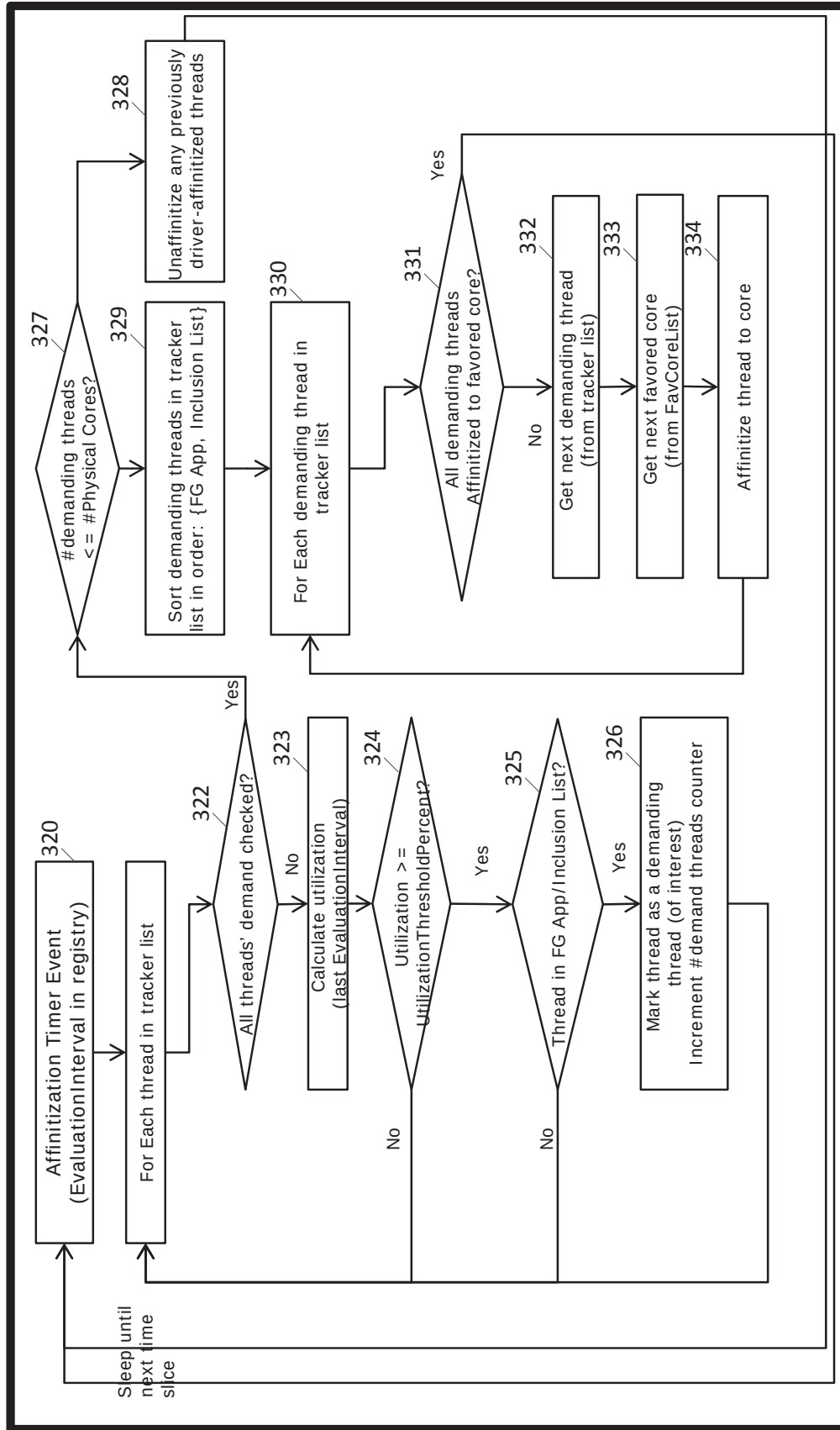


FIG. 3C

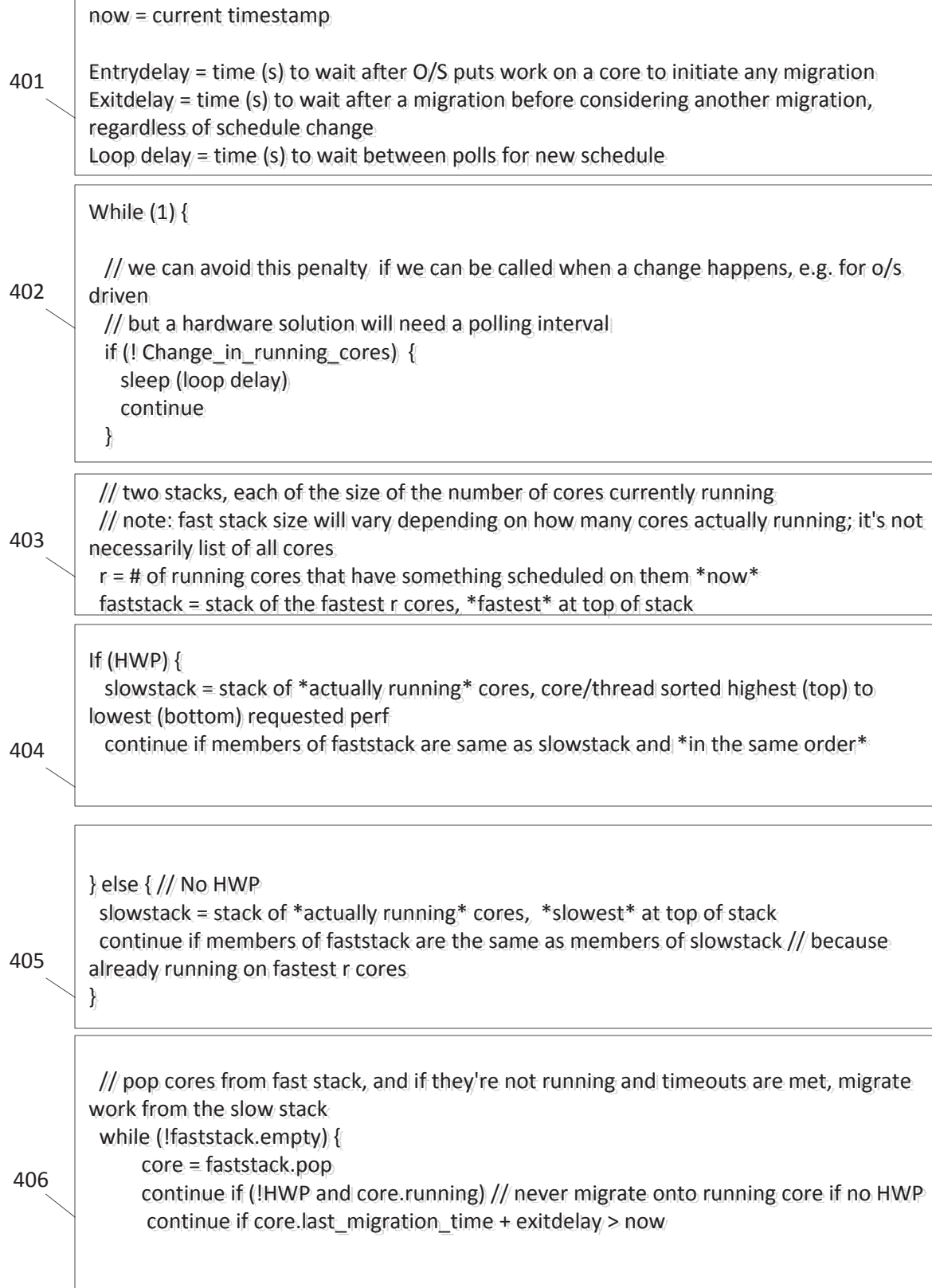


FIG. 4A

```

done = FALSE;
while (!done and !slowstack.empty) {
  victim = slowstack.pop
  continue if (victim.last_migration_time + exitdelay > now )
  continue if (victim.last_schedule_time + entrydelay > now)

  if (HWP) {
    // don't migrate if we would get more perf than requested for this workload
    continue if (core.perflevel > victim.requested_perflevel);

    // don't migrate if target core is currently running and migration to victim
    // core would put it's perf level under requirement specified by HWP
    continue if (core.running and victim.perflevel < core.requested_perflevel);

    // don't migrate if migration would exceed EPP change rate requested by
    // HWP -- compare slope to allowed rate
    continue if ((core.perflevel - victim.perflevel) / (now -
    victim.last_migration_time) > core.epp_rate))

    // don't migrate if core productivity below a threshold implying
    // migration worthwhile
    // current activity window can be o/s set or hw determined
    productivity = victim's delta-pcnt / delta-pcnt over current activity window
    continue if (productivity < productivity_threshold)

  }

  // this constraint is redundant under most conditions but avoids
  // pathological migration onto a worse core
  continue if (victim.perflevel > core.perflevel)

  // note that in HWP case if target core was running, this migration implies
  // swapping work between cores instead of merely migrating to an idle core
  migrate victim to core at time now

  done = TRUE;
}

```

407

408

409

FIG. 4B

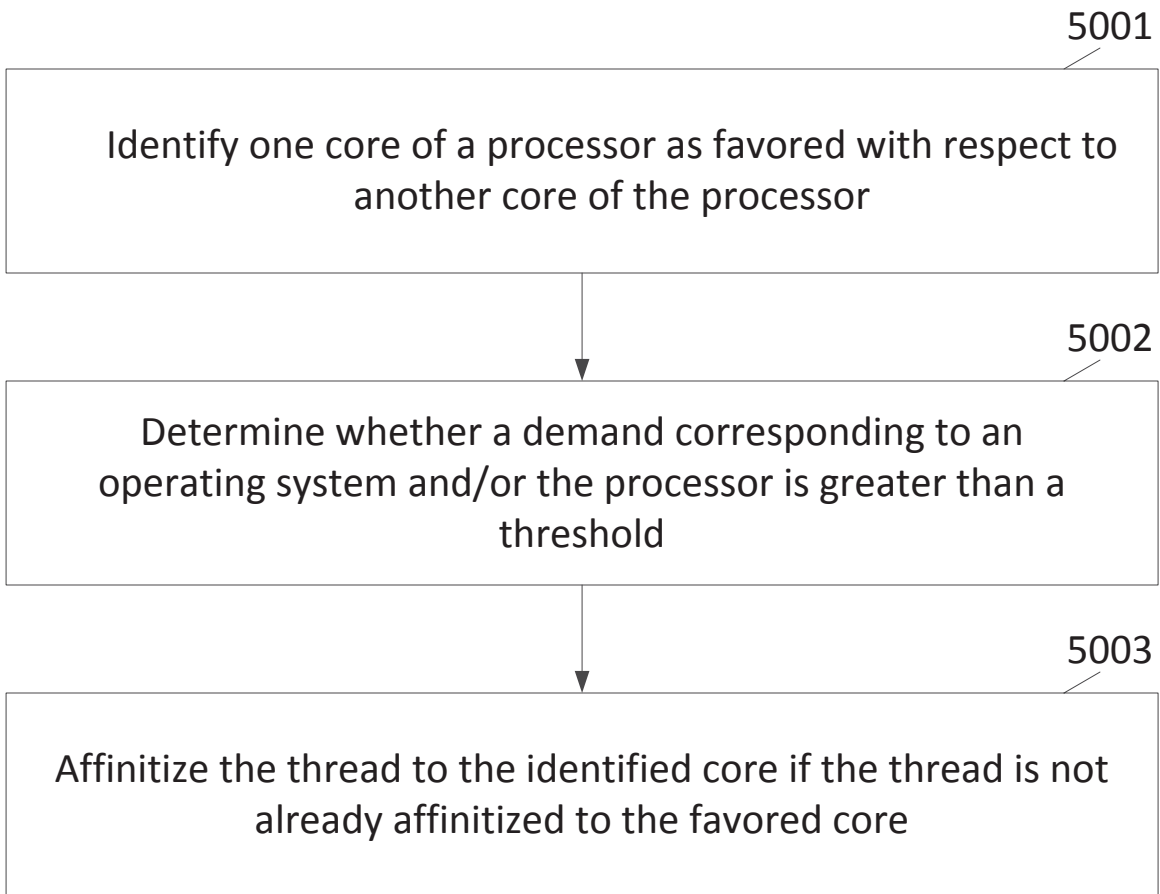


FIG. 5

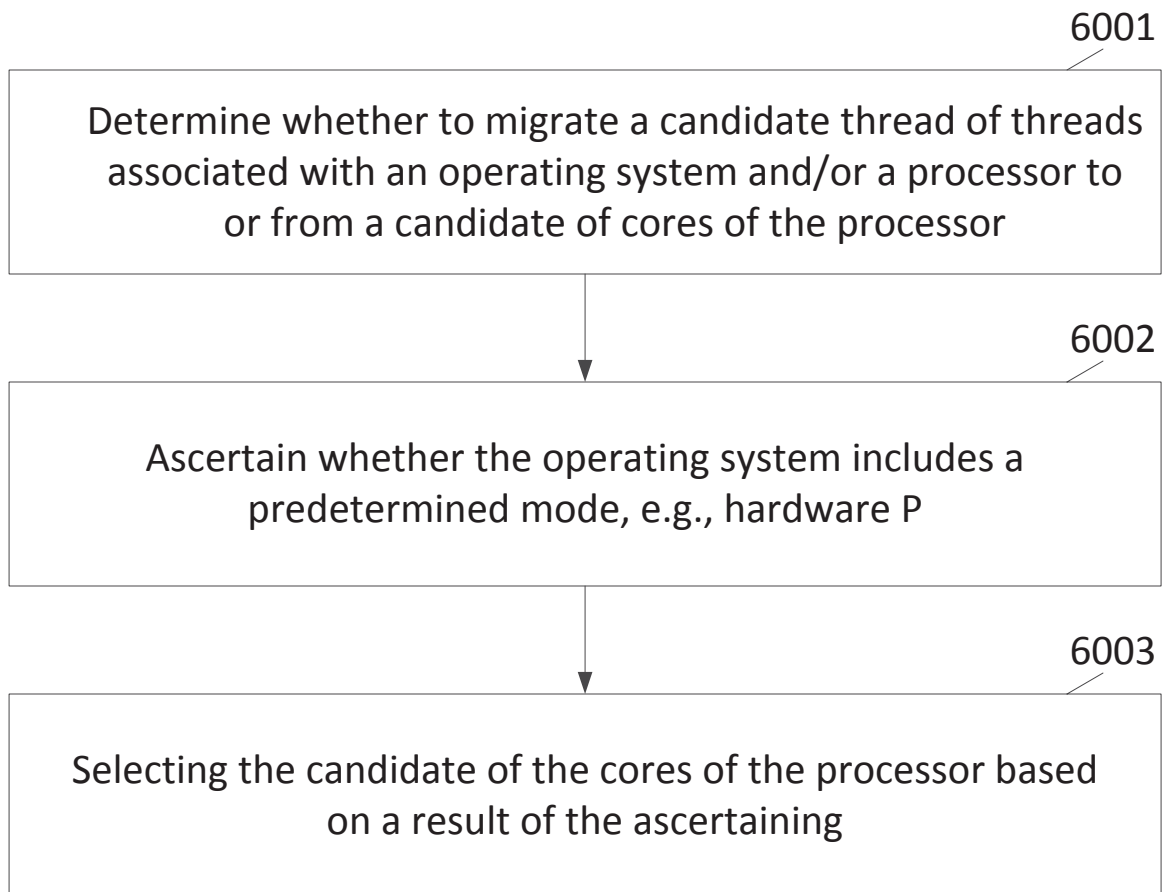


FIG. 6

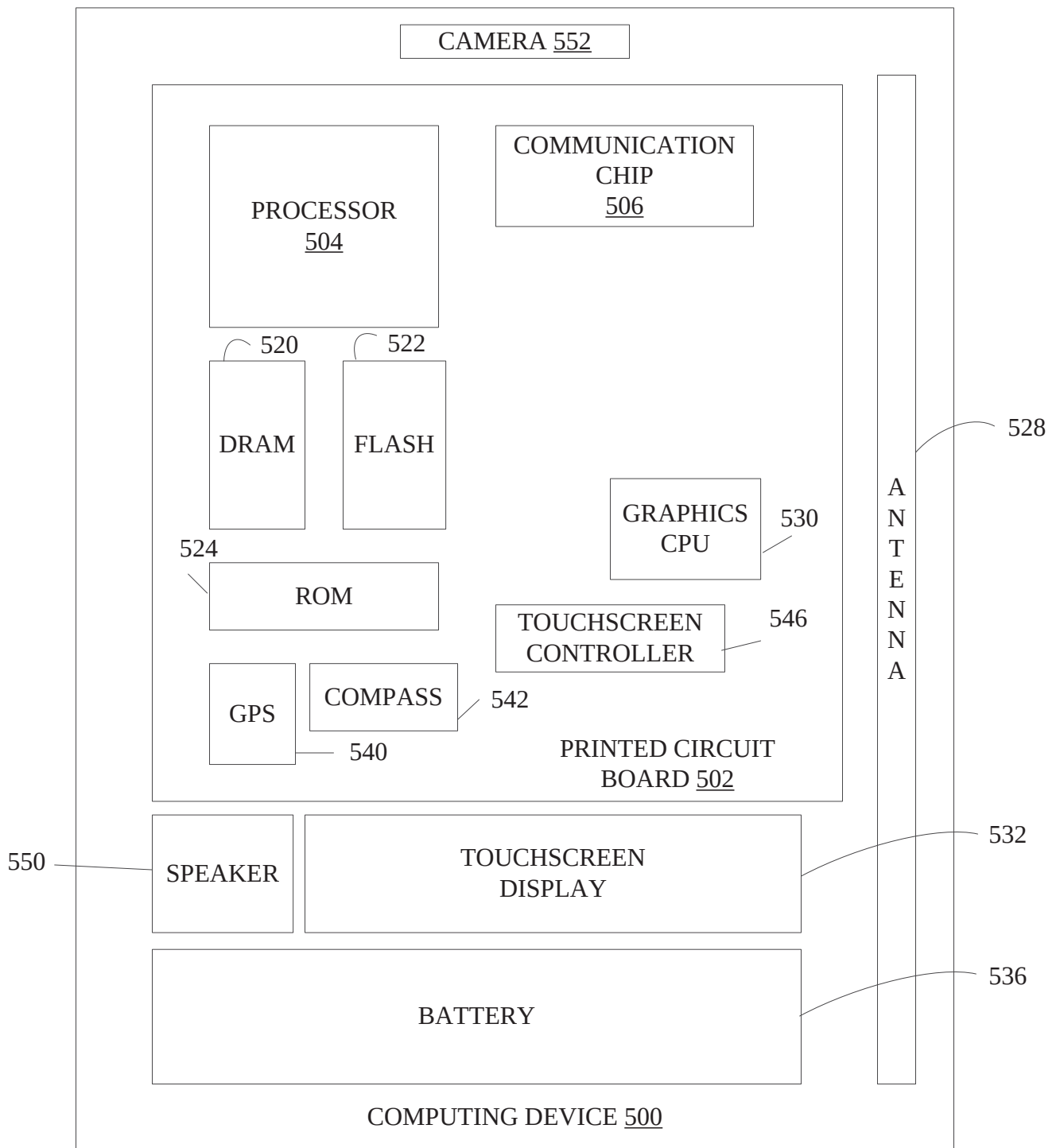


FIG. 7