Good.
Now we enter the phase where **freshers usually think they are "finally doing engineering"** — and where **most of the real damage is done if Phase-1 was weak**.

What follows is **Phase-2 exactly as practiced in avionics / defense programs**.

PHASE-2 — ENGINEERING DECOMPOSITION

(Turning system understanding into a correct hardware/firmware/mechanical architecture)

> **Phase-2 answers the question:**
> *"HOW do we build this unit so that it always behaves as Phase-1 says — even in worst-case conditions?"*

This is where **real design responsibility begins**.

---

1. WHAT PHASE-2 IS (Clear Definition)

Phase-2 = Converting system behavior into implementable engineering blocks.

Not:

- Picking parts
- Drawing schematics
- Writing firmware

But:

- Partitioning functions
- Allocating responsibilities
- Choosing architectures (not components)
- Identifying risks early

Senior truth:

> "Good architecture hides complexity and exposes guarantees."

---

2. ENTRY CRITERIA (Strict Gate)

A fresher is **not allowed** into Phase-2 unless they can:

✔ Draw Phase-1 system diagram from memory
✔ Explain all operational modes
✔ Identify critical data paths
✔ Explain failure behavior
✔ Answer "what happens if…" questions calmly

If not → Phase-1 is repeated.

---

3. PHASE-2 OBJECTIVES

By the end of Phase-2, the fresher must be able to:

1. Partition the system into **engineering domains**
2. Allocate functions to **HW / FW / SW**
3. Define **card-level architecture**
4. Understand **backplane & chassis implications**
5. Identify **design risks**
6. Prepare for **component selection (Phase-3)**

---

4. STEP-BY-STEP PHASE-2 EXECUTION

STEP 2.1 — FUNCTION → DOMAIN PARTITIONING

(The most important step)

I explain:

> "Every function must live somewhere.
> If you don't decide, chaos will."

Mandatory Domains:

Domain	Typical Responsibility
Hardware	Signal integrity, power, clocks
Firmware	Boot, low-level control
Software	Algorithms, protocols
Mechanical	Cooling, vibration
Test	Observability

Required Output #1 — FUNCTION ALLOCATION TABLE

Function	HW	FW	SW
Data acquisition	✔	✔	❌
BIT monitoring	✔	✔	✔
Protocol handling	❌	✔	✔

🚨 Freshers often over-assign to software → **danger sign**.

COMMON MISTAKE ❌

Assuming:

> "Software can fix it later."

In avionics, hardware limits software.

STEP 2.2 — CARD-LEVEL ARCHITECTURE (Core Skill)

Now we translate domains into **physical cards**.

Questions I force the fresher to answer:

Question	Why
Single card or multi-card?	Maintainability
Processor card separate?	Reuse
I/O concentrated or distributed?	Signal integrity
Redundancy at card or system level?	Safety

Required Output #2 — CARD ARCHITECTURE DIAGRAM

Example:

```
[Power Card]
      |
[Processor Card]—[XMC I/O Card]
```

```
      |
[Backplane]
```

🚨 No IC names yet
🚨 No pin numbers yet

Only **roles**.

STEP 2.3 — BACKPLANE & CHASSIS AWARENESS

I explain bluntly:

> "The backplane dictates your freedoms and your prisons."

Fresher must identify:

- Power rails provided
- Clock distribution
- Reset behavior
- Slot-to-slot communication
- Redundancy paths

Required Output #3 — BACKPLANE RESOURCE MAP

Resource	Provided by	Used by
+12V	Backplane	All cards
SysClk	Backplane	Processor
PCIe	Backplane	PPC ↔ XMC

COMMON MISTAKE ❌

Designing cards assuming:

> "Backplane will handle it."

Backplane is **not magic**.

STEP 2.4 — DATA PATH & PERFORMANCE BUDGETING

I tell them:

> "Real-time failure comes from ignoring worst-case."

Mandatory Budgets:

Parameter	Budget
Latency	ms / µs
Bandwidth	Mbps
CPU load	%
Memory	MB
Power	W

Exercise:

Follow one critical data path and answer:

- Where can it stall?
- Where can it overflow?
- Where can it corrupt?

STEP 2.5 — FAULT CONTAINMENT THINKING

Now we design for **failure isolation**.

Fresher must answer:

Fault	Detection	Reaction
Sensor failure	HW + FW	Flag
CPU hang	Watchdog	Reset
Comm loss	Timeout	Degraded mode

🚨 If they say "software will retry" → incomplete.

5. COMMON PHASE-2 FAILURES ❌

❌ Failure 1: Component-Driven Architecture

Choosing parts before architecture.

❌ Failure 2: No Power Thinking

Ignoring startup, sequencing, inrush.

❌ Failure 3: No Thermal Thinking

Assuming cooling exists.

❌ Failure 4: No Test Hooks

Designing invisible systems.

---

6. HOW SENIORS SILENTLY TEST PHASE-2 🧠

They ask:

> "Why is this function in firmware?"
> "What breaks if this card is removed?"
> "What happens during brown-out?"
> "Where do we measure this?"

Correct answers show **intentional design**, not guessing.

---

7. HOW I TRAIN FRESHERS THROUGH PHASE-2

TRAINING TASK 1 — Architecture Defense

Fresher must defend every block.

TRAINING TASK 2 — Remove a Card Scenario

"What still works?"

TRAINING TASK 3 — Worst-Case Thinking

Hot, cold, max load, fault injected.

---

8. PHASE-2 EXIT CRITERIA ✅

A fresher passes Phase-2 if they can:

✔ Draw card-level architecture
✔ Allocate functions correctly
✔ Explain backplane dependency
✔ Identify risks early
✔ Talk about fault containment

At this point, they are **allowed to touch schematics**.

---

9. WHAT PHASE-2 ENABLES NEXT

➡ **Phase-3 — Detailed Design & Component Selection**

This is where:

- PPC / ARM choice
- XMC selection
- Power ICs
- Clocking
- PCB stackup

If you want, next I can:

- Proceed to **Phase-3**
- Map Phase-2 to **real design review artifacts**
- Show **how freshers fail design reviews**
- Create **Phase-2 evaluation checklists**

Just say **"Proceed to Phase-3"**