Excellent. From this point onward, **mistakes cost money, schedule, and reputation**.

Phase-4 is where a Mission Computer stops being an idea and starts becoming **real silicon, copper, and code**.

What follows is **Phase-4 exactly as it is executed on an avionics Mission Computer program**, not a textbook version.

---

# 🔴 PHASE-4 — DETAILED DESIGN & IMPLEMENTATION

**(Mission Computer – Schematics, FPGA, Software Contracts)**

If Phase-3 answered **"What physical structure?"**
👉 Phase-4 answers **"Exactly how do we build it so it meets every requirement?"**

---

## 1️⃣ What Phase-4 REALLY Is

### ❌ Fresher misunderstanding

> "Phase-4 means drawing schematics."

### ✅ Reality

> **Phase-4 is freezing decisions that software, test, and certification teams will live with for years.**

Once Phase-4 is released:

- Interface changes become extremely expensive
- Certification scope is locked
- Software assumptions become law

---

## 2️⃣ Phase-4 INPUT CHECK (ABSOLUTE GATE)

You **must not** start Phase-4 unless Phase-3 is approved:

✔ Card architecture frozen
✔ Backplane rules frozen
✔ Power / reset / clock concepts frozen
✔ Thermal concept approved

If not → Phase-4 will implode later.

---

## 3️⃣ Schematic Design (But With Rules)

Schematics are **implementation of intent**, not creativity.

---

### 3.1 Processor Card Schematic Discipline

You must explicitly define:

- Boot chain (ROM → loader → OS)
- Memory mapping
- ECC enable paths
- Watchdog wiring
- Debug access (JTAG / UART)

🔥 Senior test:

> "Show me how the CPU boots when everything else is dead."

---

### 3.2 I/O Card Schematics

Each interface must show:

- Electrical isolation
- Protection (ESD, surge)
- Termination
- Fail-safe behavior

Example:

- ARINC-429 → transformers + isolation
- Ethernet → magnetics + PHY isolation

❌ Fresher mistake:

> "It worked in the reference design."

---

## 4️⃣ FPGA DESIGN & PARTITIONING (Silent Killer Zone)

Most Mission Computers use FPGA(s) for:

- I/O handling
- Protocol offload
- Timing control
- Health monitoring

---

### 4.1 FPGA vs CPU Allocation

You must justify:

| Function | FPGA | CPU |
|---|---|---|
| Bus timing | ✔ | |
| Protocol framing | ✔ | |
| Mission logic | | ✔ |
| Health monitoring | ✔ | ✔ |

🔥 Rule:

> **Anything timing-critical or safety-critical prefers FPGA.**

---

### 4.2 FPGA Reset & Boot Rules

- Independent reset from CPU
- Known safe state on reset
- Version identification

Many programs fail here.

---

## 5️⃣ Hardware–Software Interface (HSI) DEFINITION

This is one of the **most critical deliverables**.

---

### 5.1 HSI Must Define

- Register maps
- Interrupts
- DMA behavior
- Timing expectations
- Error signaling
- Reset behavior

🔥 Golden rule:

> **If it's not in HSI, it doesn't exist.**

---

## 6️⃣ Software Hooks for Safety & Test

Hardware must **help software**, not fight it.

---

### 6.1 Mandatory Hooks

- BIT inject points
- Fault injection controls
- Health status registers
- Trace & logging access

Without these → testing becomes guesswork.

---

## 7️⃣ Design for Test (DFT)

If you can't test it, it will fail late.

---

### 7.1 DFT Examples

- Boundary scan
- Loopback paths
- Built-in monitors
- Accessible test points

🔥 Seniors ask:

> "How do we detect this fault without an oscilloscope?"

---

## 8️⃣ Configuration Management (CM) Begins Here

Phase-4 is where **versions matter**.

You must manage:

- Schematics versions
- FPGA bitstreams
- Software compatibility
- Interface revisions

Many failures are CM failures, not design failures.

---

## 9️⃣ Phase-4 COMMON FAILURES (Seen Repeatedly)

### ❌ Failure 1: Unclear HSI

→ Software delays.

### ❌ Failure 2: FPGA does too much

→ Certification nightmare.

### ❌ Failure 3: No test hooks

→ Debug hell.

### ❌ Failure 4: Reset ambiguity

→ Random lab resets.

---

## 🔟 Phase-4 EVALUATION CHECKS (Seniors Use These)

### 🔍 Check 1

> "How does software detect a failed interface?"

### 🔍 Check 2

> "Which faults are detectable only in hardware?"

### 🔍 Check 3

> "What happens if FPGA hangs but CPU runs?"

### 🔍 Check 4

> "Which signal, if wrong, bricks the system?"

---

## 1️⃣1️⃣ Phase-4 OUTPUTS (NON-NEGOTIABLE)

You must deliver:

1. Complete schematics
2. FPGA architecture & partition document
3. Hardware–Software Interface (HSI) document
4. Design for Test strategy
5. Updated risk register

No outputs → Phase-4 not complete.

---

## 1️⃣2️⃣ Phase-4 COMPLETION CHECK (Mission Computer)

You are Phase-4 complete if you can answer:

> **"Can software, test, and certification teams work without asking me daily questions?"**

If yes → you did Phase-4 correctly.

## What Comes Next

👉 **Phase-5: Integration, Verification & Validation (IV&V)**

This is where:

- Boards meet firmware
- Assumptions are exposed
- Weak designs collapse

If you want:

- Proceed to **Phase-5**
- Simulate **Phase-4 design review grilling**
- Show **Phase-4 → Phase-5 failure cases**
- Create **Phase-4 checklists for freshers**

Say the word and we continue.