

## Dashcam SoC System Architecture Requirements (SAR)

### 1. System Context

The Dashcam SoC processes an 8-bit camera stream, stores frames to SRAM via DMA, and exposes a CSR programming model for control and status.

#### Context Diagram

```
[Camera Sensor] -> [Dashcam SoC] -> [SRAM] -> [Host (WB master or CPU)]
                  |
                  -> IRQ
```

### 2. Functional Requirements

- **REQ-CAM-001:** Accept `cam_valid`, `cam_sof`, `cam_pixel[7:0]`.
- **REQ-DMA-001:** DMA writes sequential bytes into SRAM for `DMA_LEN`.
- **REQ-CSR-001:** CSR map generated from SystemRDL with C headers + IP-XACT.
- **REQ-IRQ-001:** DMA completion triggers IRQ, clearable via `IRQ_CLEAR`.
- **REQ-WB-001:** Wishbone interconnect supports CSR + SRAM slaves.

### 3. Performance Requirements

- **REQ-PERF-001:** Target 200MHz @ 1.8V (Sky130).
- **REQ-PERF-002:** Single-clock domain in baseline design.

### 4. Reset/CDC/RDC Requirements

- **REQ-RST-001:** Async assert, sync deassert with `reset_sync`.
- **REQ-CDC-001:** No CDC in baseline; if multi-clock added, use sync/async FIFO.

### 5. Physical Design Requirements

- **REQ-PD-001:** Provide SDC and UPF for 200MHz operation.
- **REQ-PD-002:** Provide macro LEF/LIB for SRAM mapping.

### 6. Verification Requirements

- **REQ-SIM-001:** Smoke sim produces `frame_0000.ppm`.
- **REQ-STA-001:** No unconstrained paths at 200MHz target.

### 7. Flowchart: System Bring-Up

```
[Power-on Reset]
  |
  v
[Program DMA_BASE/DMA_LEN]
  |
  v
[Enable Camera + Start DMA]
  |
  v
[Wait IRQ or Poll DMA_STATUS]
  |
```

v  
[Read SRAM Frame]

## 8. Traceability

- RTL: `top/dashcam_top/rtl/dashcam_soc_top.sv`
- CSR spec: `ips/csr_regs/dashcam_csr.rdl`
- Address map: `docs/specs/address_map/address_map.md`