**Safely Riding a Car — Complete Solution (simple, step-by-step)**

**Step 1 — Understand & Define the Problem (analysis — in plain English)**

We need a logic circuit that switches an **alarm line (active-LOW)** on whenever the car is started **and** at least one front seat is occupied **and** that occupant’s seatbelt is unfastened.

Inputs:

* DRIV — driver present (1 = yes)
* PASS — passenger present (1 = yes)
* IGN — ignition on (1 = yes)
* BELTD̅ — driver belt signal, **active-LOW** (0 = unfastened, 1 = fastened)
* BELTP̅ — passenger belt signal, **active-LOW** (0 = unfastened, 1 = fastened)

Output:

* ALARM̅ — alarm line, **active-LOW** (0 = alarm ON, 1 = alarm OFF)

Rule in one sentence:

* If IGN = 1 and (driver seated & driver unfastened OR passenger seated & passenger unfastened), then ALARM̅ = 0 (ON). Otherwise ALARM̅ = 1 (OFF).

**Step 2 — Organise & Describe the Data (short table)**

| **Signal** | **Meaning** | **Active when** |
| --- | --- | --- |
| DRIV | Driver seated | 1 = seated |
| PASS | Passenger seated | 1 = seated |
| IGN | Ignition on | 1 = started |
| BELTD̅ | Driver belt (active-LOW) | 0 = unfastened, 1 = fastened |
| BELTP̅ | Passenger belt (active-LOW) | 0 = unfastened, 1 = fastened |
| ALARM̅ | Alarm output (active-LOW) | 0 = alarm ON, 1 = alarm OFF |

Operational notes:

* If IGN = 0 → alarm must be OFF (ALARM̅ = 1) no matter seat states.
* If IGN = 1 → evaluate seat occupancy and belt state.

**Step 3 — Plan the Solution (design the algorithm)**

**3.1 Algorithm (plain English)**

1. Read inputs: IGN, DRIV, PASS, BELTD̅, BELTP̅.
2. If IGN = 0 → set ALARM̅ = 1 (OFF) and stop.
3. Convert belt signals to “unfastened” flags:
   * D\_unfast = NOT(BELTD̅) (1 when driver unfastened)
   * P\_unfast = NOT(BELTP̅) (1 when passenger unfastened)
4. If (DRIV AND D\_unfast) OR (PASS AND P\_unfast) is true → ALARM̅ = 0 (ON). Else ALARM̅ = 1 (OFF).

**3.2 Compact Boolean (concept)**

Let D\_un = ¬BELTD̅, P\_un = ¬BELTP̅. Internal alarm request:

REQ = IGN ⋅ ( DRIV⋅D\_un + PASS⋅P\_un )

Output (active-LOW):

ALARM̅ = ¬REQ

**3.3 Boolean expression (SOP, easy form)**

Expressed fully:

ALARM̅ = ¬ [ IGN ⋅ ( DRIV ⋅ ¬BELTD̅ + PASS ⋅ ¬BELTP̅ ) ]

Alternative (expanded / POS-friendly) using De Morgan:

ALARM̅ = ¬IGN + (¬DRIV + BELTD̅) ⋅ (¬PASS + BELTP̅)

(That form can be handy if you prefer OR/AND blocks or want to implement with NAND/OR variants.)

**3.4 Pseudocode**

read IGN, DRIV, PASS, BELTD\_bar, BELTP\_bar # BELT\*\_bar are active-LOW

if IGN == 0:

ALARM\_bar = 1 # alarm off

else:

D\_unfast = (BELTD\_bar == 0)

P\_unfast = (BELTP\_bar == 0)

if (DRIV == 1 and D\_unfast) or (PASS == 1 and P\_unfast):

ALARM\_bar = 0 # alarm on

else:

ALARM\_bar = 1 # alarm off

**3.5 Flowchart**

**Step 4 — Implement the Solution**

**4.1 Gate-level (logic circuit description — simple)**

* Invert BELTD̅ → D\_un (NOT gate)
* Invert BELTP̅ → P\_un (NOT gate)
* AND DRIV with D\_un → A1 (AND gate)
* AND PASS with P\_un → A2 (AND gate)
* OR A1 and A2 → S (OR gate)
* AND S with IGN → REQ (AND gate)
* NOT REQ → ALARM̅ (NOT gate)

Gate count: 3 NOTs (2 for belts, 1 final), 3 ANDs, 1 OR (total small and simple).

**4.2 Python Code — Version 1 (clear)**

def alarm\_bar\_v1(DRIV, PASS, IGN, BELTD\_bar, BELTP\_bar):

# Inputs: integers 0 or 1. BELT\*\_bar: 0 = unfastened, 1 = fastened

if IGN == 0:

return 1 # alarm OFF (active-high value 1)

d\_unfast = 1 if BELTD\_bar == 0 else 0

p\_unfast = 1 if BELTP\_bar == 0 else 0

if (DRIV == 1 and d\_unfast == 1) or (PASS == 1 and p\_unfast == 1):

return 0 # alarm ON (active-low)

else:

return 1 # alarm OFF

**4.3 Python Code — Version 2 (compact boolean)**

def alarm\_bar\_v2(DRIV, PASS, IGN, BELTD\_bar, BELTP\_bar):

if not IGN:

return 1

d\_un = (BELTD\_bar == 0)

p\_un = (BELTP\_bar == 0)

alarm\_request = (DRIV and d\_un) or (PASS and p\_un)

return 0 if alarm\_request else 1

**Step 5 — Test and Refine (simple & practical)**

**5.1 Test cases (small table)**

| **Case** | **DRIV** | **PASS** | **IGN** | **BELTD̅** | **BELTP̅** | **Expected ALARM̅** | **Why** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | x | x | 1 | Ignition off → alarm off |
| 2 | 1 | 0 | 1 | 0 | x | 0 | Driver seated & unfastened → alarm ON |
| 3 | 1 | 0 | 1 | 1 | x | 1 | Driver seated & fastened → alarm OFF |
| 4 | 0 | 1 | 1 | x | 0 | 0 | Passenger seated & unfastened → alarm ON |
| 5 | 1 | 1 | 1 | 0 | 0 | 0 | Both unfastened → alarm ON |
| 6 | 0 | 0 | 1 | x | x | 1 | No occupants → alarm OFF |

(x = don’t care)

Run these through alarm\_bar\_v2 — outputs should match the “Expected” column.

**5.2 How to debug if a test fails**

* Check signal polarity: BELTD̅ and BELTP̅ are **active-LOW** (0 means unfastened). If your wiring or reading flips this, results invert.
* Confirm IGN reading: if ignition reading is wrong, alarm will behave incorrectly.
* Check occupancy switches (DRIV, PASS) for bounce or wrong wiring.
* In code, add print/log lines to show intermediate flags (d\_un, p\_un, alarm\_request).

**5.3 Refinements / practical notes**

* In hardware, **debounce** seat switches or use Schmitt inputs to avoid flicker while someone moves.
* Consider a short startup delay after ignition (0.5–1 s) before evaluating belts to avoid false alarms at power on.
* If you must use an active-HIGH alarm module, invert accordingly or remove the final inverter and wire to an active-HIGH input.
* If you want a visual LED plus buzzer: drive buzzer from ¬ALARM̅ (i.e., alarm\_request) so logic matches device polarity.

**Quick summary (one-liner)**

Set ALARM̅ = 0 when the car is on (IGN = 1) and any occupied front seat has an unfastened belt (driver or passenger); otherwise keep ALARM̅ = 1.

If you want, I can now:

* Produce a **full 32-row truth table** (all combinations) and the simplified SOP terms, or
* Generate a **Draw.io XML** for the flowchart or gate diagram you can import, or
* Show a **short test script** that loops through all inputs and prints results for you to paste into IDLE.

Which of those would you like next?