**Tutorial Week 3 – Car Seatbelt Alarm System**

### **Step 1 – Understand and Define the Problem (Analyse)**

The car safety system should activate an alarm when:

1. The ignition is ON (IGN = 1)
2. Either seat is occupied and the corresponding seatbelt is unfastened

**Key input signals:**

* DRIV – Driver present (1 = yes, 0 = no)
* PASS – Passenger present (1 = yes, 0 = no)
* IGN – Ignition ON (1 = yes, 0 = no)
* BELT\_D̅ – Driver seatbelt unfastened (1 = unfastened, 0 = fastened)
* BELT\_P̅ – Passenger seatbelt unfastened (1 = unfastened, 0 = fastened)

**Output signal:**

* ALARM – Active-LOW (0 = alarm ON, 1 = alarm OFF)

### **Step 2 – Organise and Describe the Data**

**Logical conditions:**

* **Alarm ON when:** IGN = 1 AND (DRIV=1 AND BELT\_D̅=1 OR PASS=1 AND BELT\_P̅=1)

### 

### **Step 3 – Plan the Solution (Design the Algorithm)**

#### **3.1 Algorithm in Plain English**

1. Check if ignition is ON.
2. If ignition is OFF, the alarm stays OFF.
3. If ignition is ON, check the driver seat:
   * If the driver is present AND the belt is unfastened → alarm ON.
4. Check passenger seat:
   * If a passenger is present AND the belt is unfastened → alarm ON.
5. Otherwise, alarm OFF.

| **Symbol** | **Description** |
| --- | --- |
| **DRIV** | Driver present (1 = Yes, 0 = No) |
| **PASS** | Passenger present (1 = Yes, 0 = No) |
| **IGN** | Ignition ON (1 = Yes, 0 = No) |
| **BELT\_D̅** | Driver seatbelt unfastened (1 = Unfastened, 0 = Fastened) |
| **BELT\_P̅** | Passenger seatbelt unfastened (1 = Unfastened, 0 = Fastened) |
| **ALARM** | Alarm output (0 = Alarm ON, 1 = Alarm OFF) |

### 

### 

### 

### 

### 

### **3.2 Truth Table**

| **DRIV** | **PASS** | **IGN** | **BELT\_D̅** | **BELT\_P̅** | **ALARM** |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |

#### 

#### 

#### 

#### **3.3 Boolean Expression (SOP)**

For **ALARM\_ON** (active-HIGH version):

ALARM\\_ON = IGN \cdot DRIV \cdot BELT\\_D̅ + IGN \cdot PASS \cdot BELT\\_P̅

For **ALARM** (active-LOW output):

ALARM = \overline{IGN \cdot DRIV \cdot BELT\\_D̅ + IGN \cdot PASS \cdot BELT\\_P̅}

**3.4 Pseudocode**

IF IGN = 1 THEN

IF (DRIV = 1 AND BELT\_D̅ = 1) OR (PASS = 1 AND BELT\_P̅ = 1) THEN

ALARM = 0 // ON

ELSE

ALARM = 1 // OFF

ENDIF

ELSE

ALARM = 1 // OFF

ENDIF

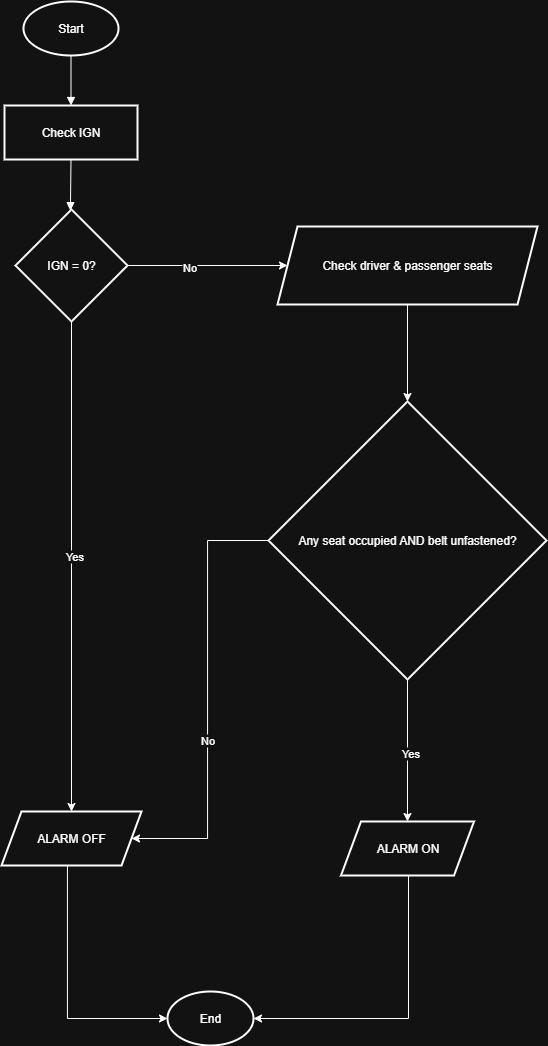
#### 

#### 

#### 

#### **3.5 Flowchart**

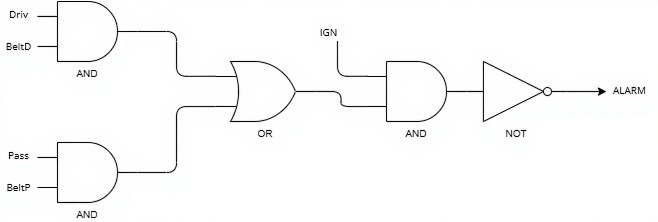
* Start
* Check IGN
  + If 0 → ALARM OFF → End
  + If 1 → Check driver & passenger seat conditions
  + If any seat occupied AND the belt is unfastened → ALARM ON
  + Else → ALARM OFF
* End



### **Step 4 – Implement the Solution**

#### **4.1 Logic Circuit**

* 2 × 3-input AND gates
* 1 × 2-input OR gate
* 1 × NOT gate (to make ALARM active-LOW)

****

**4.2 Python Code**

# Inputs (1 or 0)

DRIV = int(input("Driver present? (1/0): "))

PASS = int(input("Passenger present? (1/0): "))

IGN = int(input("Ignition ON? (1/0): "))

BELT\_D\_bar = int(input("Driver belt unfastened? (1/0): "))

BELT\_P\_bar = int(input("Passenger belt unfastened? (1/0): "))

# Logic

if IGN == 1 and ((DRIV == 1 and BELT\_D\_bar == 1) or (PASS == 1 and BELT\_P\_bar == 1)):

ALARM = 0 # ON

else:

ALARM = 1 # OFF

print("ALARM =", ALARM)

**Step 5 – Test and Refine**

### **5.1 Purpose**

* To confirm that the logic circuit, Boolean expression, and Python code all produce the same, correct output.
* To catch any wiring, logic, or coding mistakes.
* To refine the design for efficiency or clarity if needed.

### **5.2 Testing the Logic Circuit**

1. Open your logic simulator (LogicCircuit app from your brief or Logisim).
2. Build the circuit using:
   * Two 3-input AND gates (one for IGN·DRIV·BELT\_D\_BAR, one for IGN·PASS·BELT\_P\_BAR).
   * One OR gate (to combine the AND outputs).
   * One NOT gate (to make ALARM active-LOW).
3. Label every input and output:
   * Inputs: DRIV, PASS, IGN, BELT\_D\_BAR, BELT\_P\_BAR.
   * Output: ALARM.
4. Test all input combinations:
   * Use the truth table from Step 3.2 as your checklist.
   * For each combination, toggle the switches and check if ALARM matches the table.
5. If a mismatch occurs:
   * Checking gate connections and ensuring correct inputs.
   * Verifying you didn’t confuse BELT\_D\_BAR (1 = unfastened) with belt fastened signals.
   * Ensuring the NOT gate is on the final output and not accidentally inside one of the inputs.

**5.3 Testing the Python Code**

* Running version1.py in your Python environment.
* Input each set of values from the truth table.
* Checking if the printed ALARM matches the truth table.

**5.4 Refining the Solution**

**Circuit refinement:**

* Ensuring gate inputs are at their minimal possible number (no unnecessary inputs).

**Code refinement:**

* Ensuring the variable names are clear and match the circuit.
* Removing redundant parentheses or conditions.

**Documentation refinement:**

* Updating the truth table if any logic changes.
* Save screenshots of the working circuit simulation with test cases.

### **5.5 Final Verification Checklist**

The circuit matches the truth table for all 32 input combinations.  
 Both Python versions give identical outputs for all inputs.  
 Boolean expression matches implemented circuit logic.  
 Documentation (truth table, pseudocode, Boolean expression, flowchart) is consistent.