Hazards in Digital Logic Design

Think of them like hiccups in digital circuits - the logic is correct, but delays in different paths cause sudden, incorrect spikes.

Glitch

A glitch is a momentary, unintended change or spike in the output of a digital logic circuit, usually caused by differences in signal propagation delays.

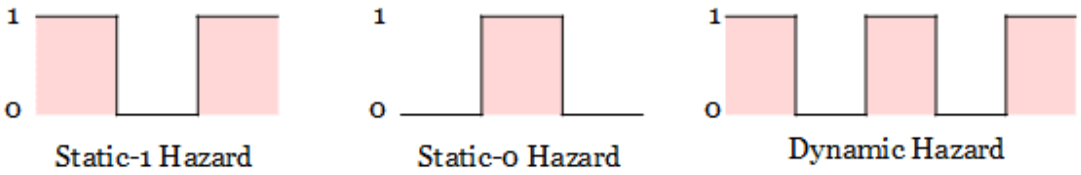
- Glitches are **temporary pulses** (high or low) that shouldn't occur.
- They occur due to **timing mismatches**, not because of incorrect logic design.
- They are **visible on waveforms** and can affect sensitive circuits like counters, memory, or communication systems.

Hazards

A **hazard** is a **design flaw or condition** in a logic circuit where the output **may temporarily go wrong** due to **unequal path delays**, even though the Boolean function is logically correct.

- Hazards are **predictable** using **Karnaugh Maps** or circuit analysis.
- They can **cause glitches** if not handled.

Type of hazard



Feature	Hazard	Glitch
Definition	A potential condition in logic design that may cause a glitch.	An actual unwanted pulse in output caused by a hazard or delay.
Type	It's a theoretical or design-level issue (predictable using K-maps).	It's a practical, observed behavior in real circuit waveforms.
Cause	Unequal delays in logic paths + improper logic coverage.	Triggered by hazards or race conditions; due to delay mismatches**.**
Timing	Happens during input transitions.	Occurs briefly and unexpectedly—microseconds or nanoseconds.

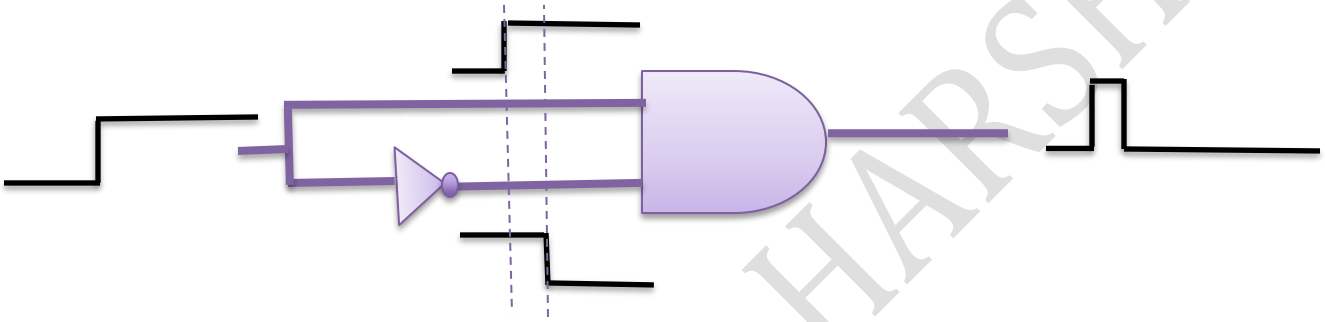
Hazards are the cause, and glitches are the effect.

➢ Static Hazard

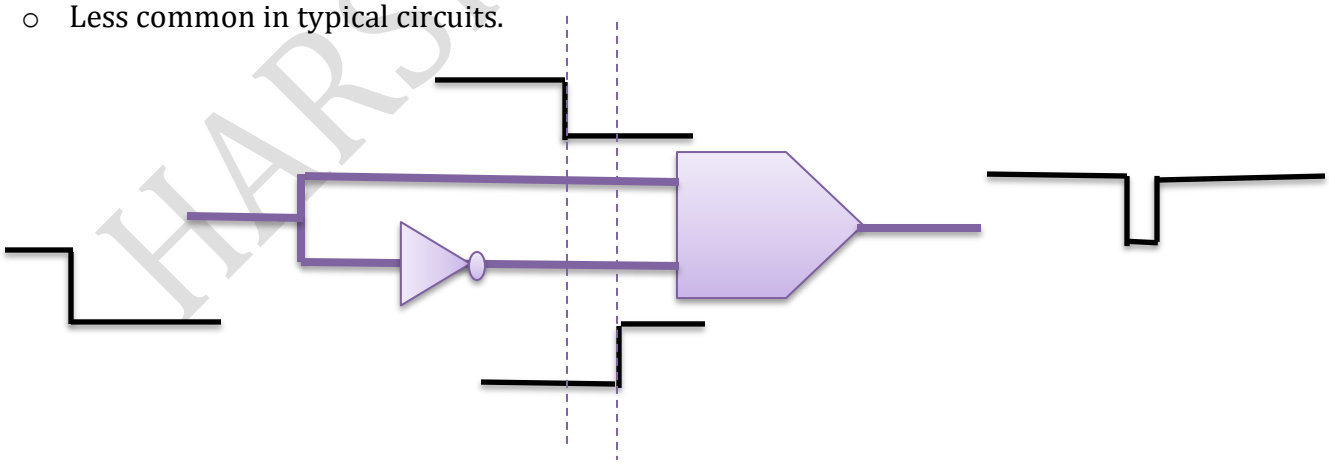
A **static hazard** occurs in a combinational logic circuit when the output is **expected to remain steady** (either logic 0 or logic 1), but due to **different propagation delays** in logic paths, the output **briefly changes** (glitches) during a transition, even though the final output remains correct.

Types of Static Hazards

- **Static-1 Hazard:**
  - The output **should remain 1**.
  - But due to unequal delays, it **temporarily drops to 0** before returning to 1.
  - This is **the most common** type of static hazard



- Static-0 Hazard:
  - The output **should remain 0**.
  - But it **momentarily rises to 1** before returning to 0.
  - Less common in typical circuits.



When is Static Hazard Important?

- **Safety-Critical Systems:** Pacemakers, automotive ECUs, industrial automation.
- **Clock-sensitive Designs:** Hazard can trigger unintended clocking events.
- **Glitch-sensitive Inputs:** Like enabling a flip-flop or triggering control lines.

➢ Dynamic Hazard

A **dynamic hazard** occurs when the output of a logic circuit is **supposed to change only once** (from 0 to 1 or 1 to 0), but due to complex delays in the circuit, it changes **multiple times before settling**.

- **Dynamic hazards** typically happen in **circuits with multiple logic levels** and **longer paths**.
- Multiple gates have to switch in sequence, and the **different gate delays** can cause the output to flutter.

Even though the Boolean expression and truth table are correct, the output signal is **unstable during transitions** because different parts of the circuit reach their final value **at different times**.

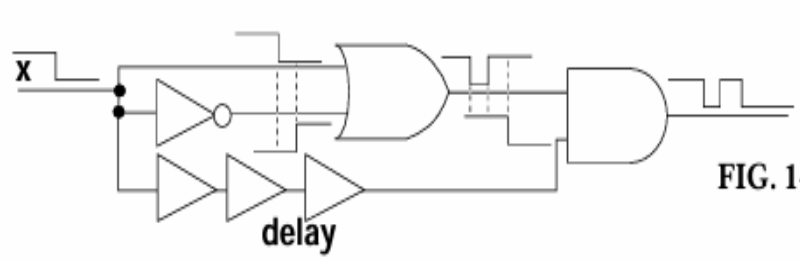


FIG. 1-5 The basic dynamic hazard circuit with an imbedded static-1 hazard.

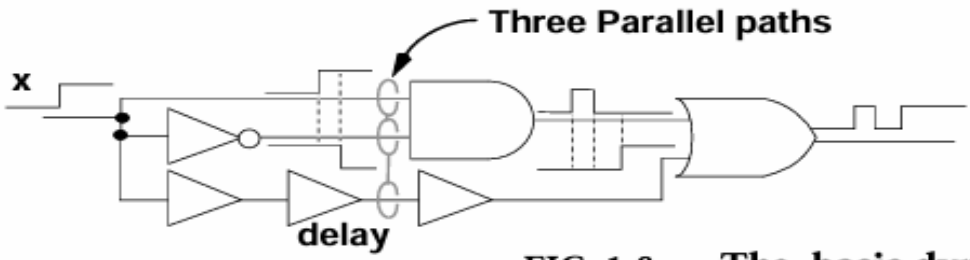


FIG. 1-6 The basic dynamic hazard circuit with an imbedded static-0 hazard.

Scenario	Why Dynamic Hazard Matters
High-speed logic	Can cause false triggering or glitches
Edge-triggered flip-flops	Glitches can be latched as valid inputs
FSMs and control logic	May jump to invalid or dangerous states
Safety-critical applications	Even rare glitches can lead to failure
Mixed-signal or async input domains	Can amplify metastability and uncertainty

Dynamic hazards are like a traffic jam during a signal change. Too many cars (signals) trying to move, some go too fast, some go too slow — and you get chaos at the intersection (output)!

➢ Eliminating Hazards

- Using Redundant Terms:

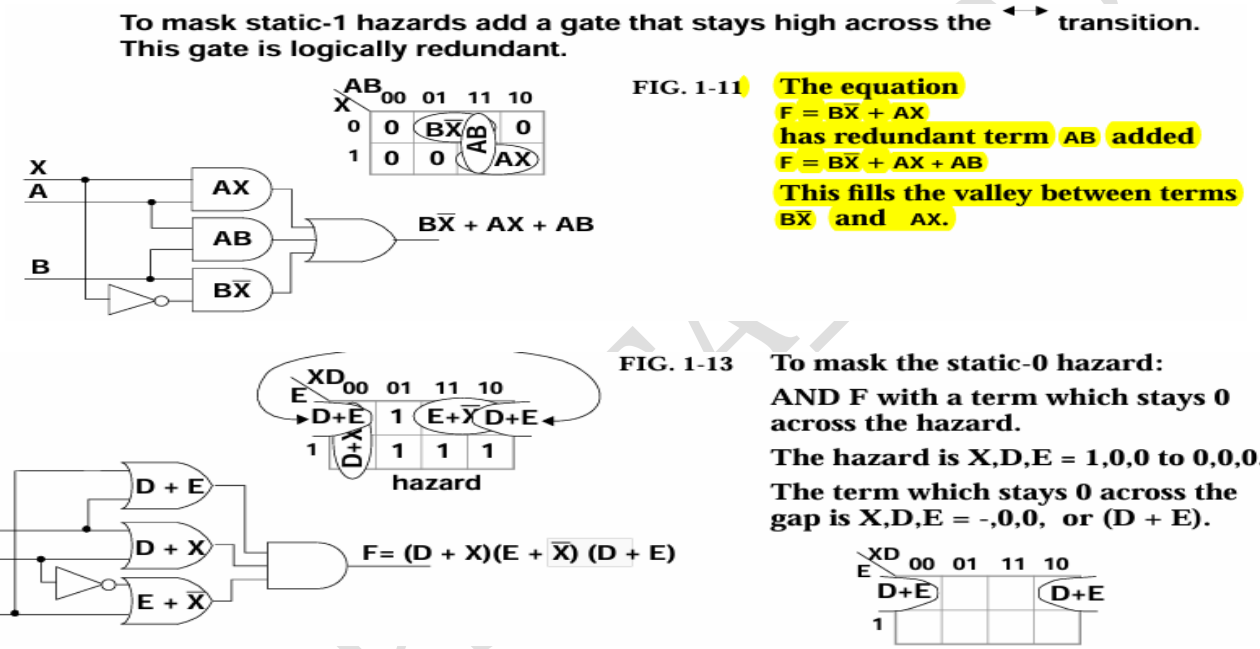
Add **extra product terms** (redundant logic) to the Boolean expression that **don't change the logical outcome** but **cover the transition paths** that cause hazards.

□ Why It Works:

If two minterms are not adjacent in the Karnaugh Map (K-map) and a transition occurs between them, the output may temporarily go to 0. A redundant term **bridges** this gap, preventing the glitch.

□ How To Do It:

- **Draw the K-map** of the original function.
- Identify minterms that are logically close but not adjacent.
- Add a new group that **overlaps** those two.
- Include the corresponding product term in your final expression.



Redundant terms = insurance policies for digital circuits.

- Proper Gate-Level Design (Path Delay Matching)

Design the circuit at the **gate level** such that the **delays in signal paths are matched**, minimizing the risk of one signal arriving too early or too late.

Why It Works:

Hazards arise when some inputs change and certain signals take longer to propagate. Ensuring that all signals reach their destination **in sync** can eliminate hazards.

□ How To Do It:

- Use **delay elements** or **buffer gates** to equalize paths.
- Ensure the **critical paths** are analyzed and balanced.