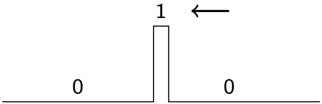
# Hazards

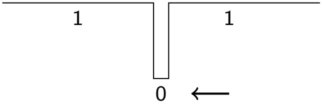
- A *hazard* is a momentary *unwanted* switching transient at a logic function's output; i.e., a hazard produces a *glitch* at the output of a circuit.
- Hazards result due to an unequal propagation delay along different paths in a combinational circuit from an input to the logic function's output.
- Hazards may be either *static* or *dynamic* and are classified by the type of glitch they may produce.

# Static hazards

- There are two types of static hazards — static-0 hazards and static-1 hazards.
- These happen when circuit inputs change and the logic function value should **not** change, but it does temporarily.
- Static-0 hazard:

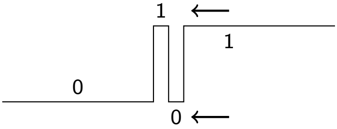
  The 1 should not have happened. The "width" or amount of time the logic function output is 1 is very narrow. The logic value should have been constant at 0.
- Static 1 hazard:

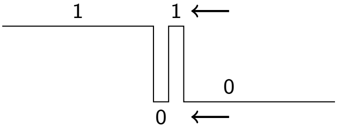
  The 0 should not have happened. The "width" or amount of time the logic function output is 0 is very narrow. The logic function output should have been at a constant value of 1.

# Dynamic hazards

- A dynamic hazard occurs when inputs change and the logic function output is expected to change from 0 → 1 or from 1 → 0.
- However, in making the transition, the logic function output "flips" one or more times.
- Example... signal changing 0 → 1...



- Example... signal changing 1 → 0...

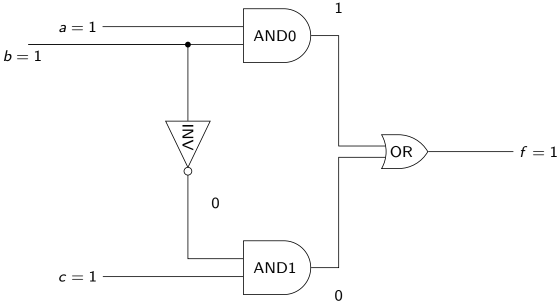


# Concerns with hazards

- In a sequential circuit, hazards are not really a concern since we have a clock period for signals to stabilize prior to the active clock edge causing the flip flops to update.
- In an asynchronous circuit, however, hazards can cause terrible problems — the glitch caused by the hazard can propagate back in the circuit and cause the circuit to malfunction (e.g., enter into an incorrect state).
- In some cases, we can mask or prevent hazards from causing glitches.

# Illustration of static hazards

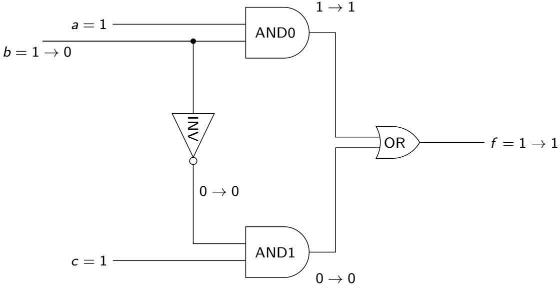
- Consider the following circuit... Assume that every logic gate has 1 unit of delay. Further, assume that input $b$ changes from $1 \rightarrow 0$ at time 0.



- Diagram shows the value of all logic signals prior to $b$ changing.

# Illustration of static hazards

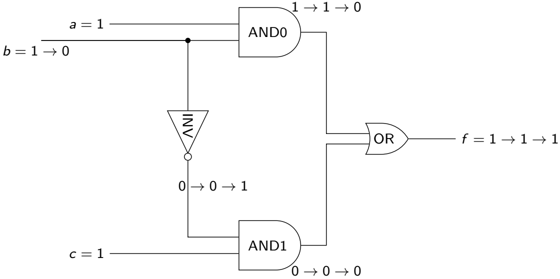
- Signal values at time 0... Signal $b$ changes from $1 \rightarrow 0$.



- Notice... The outputs of INV and AND0 do **not change immediately** even though logically they will change.

# Illustration of static hazards

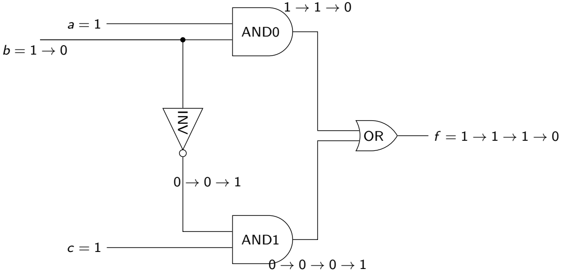
- Signal values at time 1...



- Notice... The outputs of INV and AND0 have changed at time 1. It's now the case that the output of AND1 does **not change immediately** even though logically it will change.

# Illustration of static hazards

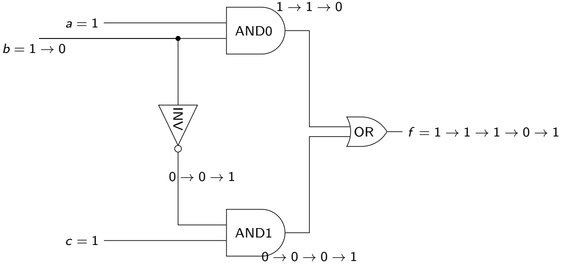
- Signal values at time 2...


$a = 1$
$b = 1 \rightarrow 0$
$c = 1$
AND0
$1 \rightarrow 1 \rightarrow 0$
INV
$0 \rightarrow 0 \rightarrow 1$
AND1
$0 \rightarrow 0 \rightarrow 0 \rightarrow 1$
OR
$f = 1 \rightarrow 1 \rightarrow 1 \rightarrow 0$

- Notice... The output of AND1 has changed. It is also the case that the output of OR changed to zero at time 2. This was because at time 1, both its inputs were 0. Note now, however, that the inputs to OR imply that its output should be 1, but the output does **not change immediately**.

# Illustration of static hazards

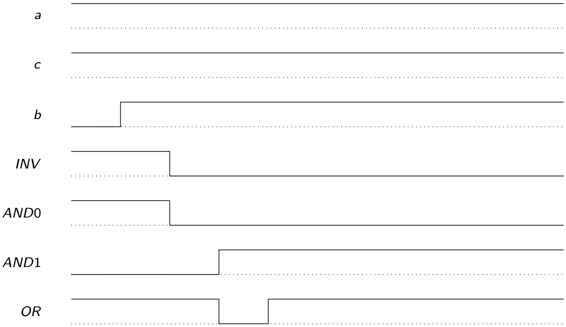
- Signal values at time 3...



- Notice... The output of OR has changed to 1. It should be clear that the OR gate output which was initially 1 should have remained 1, but did temporarily change to 0.
- **So we have encountered a static** 1 **hazard.**

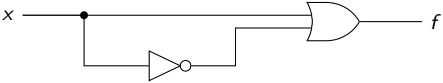
# Illustration of static hazards

- Signals drawn in time...

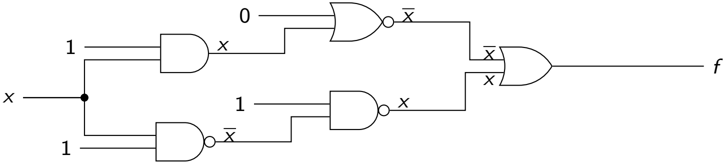


# Basic static-1 hazards

- We will consider static-1 hazards caused due to a change in a single input variable.
- Characterized by the following:



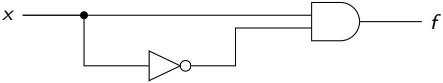
- In other words, we have a circuit in which there are two parallel paths from input $x$ and one of these paths is inverted. The paths reconverge at an OR gate.
- Here $f = x + \overline{x}$ and, due to delays, $x$ and $\overline{x}$ need to be considered as separate signals. Therefore, it is possible that OR gate will see 0 at both inputs for a brief moment in time.
- Potential static-1 hazard embedded in a circuit...



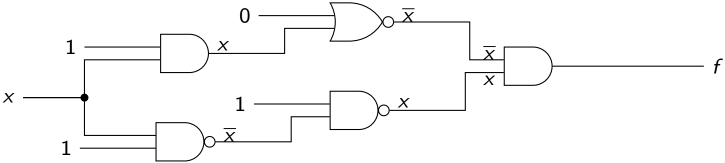
Still see $x$ and $\overline{x}$ at the input to the OR gate.

# Basic static-0 hazards

- We will consider static-0 hazards caused due to a change in a single input variable.
- Characterized by the following:



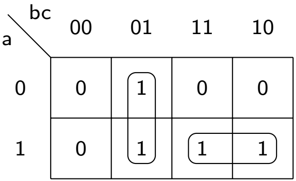
- In other words, we have a circuit in which there are two parallel paths from input $x$ and one of these paths is inverted. The paths reconverge at an AND gate.
- Here $f = x\overline{x}$ and, due to delays, $x$ and $\overline{x}$ need to be considered as separate signals. Therefore, it is possible that AND gate will see 1 at both inputs for a brief moment in time.
- Potential static-0 hazard embedded in a circuit...



Still see $x$ and $\overline{x}$ at the input to the AND gate.

# Static-1 hazards and sum-of-products (redundant terms)

- Sum-of-products expressions may potentially have static-1 hazards due to one input variable changes.
- Sum-of-products expressions will not have static-0 hazards or dynamic hazards.
- Masking hazards in sum-of-products expressions due to single input variable changes is pretty straightforward.
- Example... $f = ab + \overline{b}c$... Its Karnaugh map...

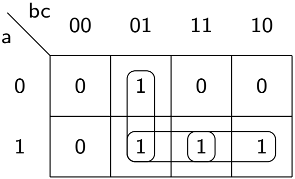


| a \ bc | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |

- The identified product terms are how the function is implemented.
- **Observe that when $b = 1 \rightarrow 0$, that we "jump" from one product term to another.** Further, at any given time, there is only 1 product term responsible for holding $f = 1$.

# Static-1 hazards and sum-of-products (redundant terms)

- Introduce another product term without changing the function.

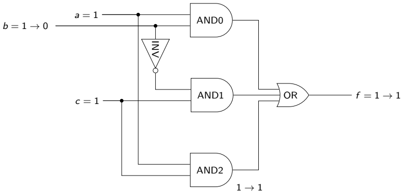


| a \ bc | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |

- Here, we can write $f = ab + \overline{b}c + ac$.
- The additional (redundant) product term is independent of input $b$ changing. This additional product term will be 1 while $b = 1 \rightarrow 0$ and will hold $f = 1$.

# Static-1 hazards and sum-of-products (redundant terms)

- Circuit...



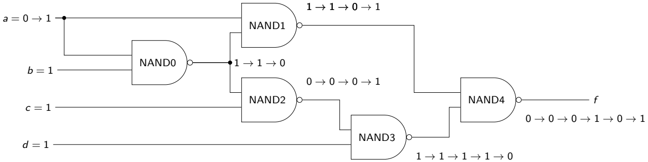
- To fix static-1 hazards in sum-of-products expressions we add redundant product terms to ensure that a single input variable changing does not cause the input variable and its complement to be the only inputs to the OR gate. The redundant product term "holds" the output at 1 and prevents the static-1 hazard.

# Static-0 hazards and product-of-sums (redundant terms)

- Product-of-sums expressions may potentially have static-0 hazards due to one input variable changes.
- Product-of-sums expressions will not have static-1 hazards or dynamic hazards.
- Masking hazards in product-of-sums expressions due to single input variable changes is pretty straightforward. Add redundant sum terms which are independent of the single input variable that causes the hazard.
- Example... $f = (a + \overline{b})(b + c)$... This implementation has a static-0 hazard when $a = 0$, $c = 0$ and $b$ changes from $1 \rightarrow 0$. This hazard is masked by adding the redundant sum term $(a + c)$ and use $f = (a + \overline{b})(b + c)(a + c)$.
- To fix static-0 hazards in product-of-sums expressions we add redundant sum terms to ensure that a single input variable changing does not cause the input variable and its complement to be the only inputs to the AND gate. The redundant sum term "holds" the output at 0 and prevents the static-0 hazard.

# Illustration of dynamic hazards

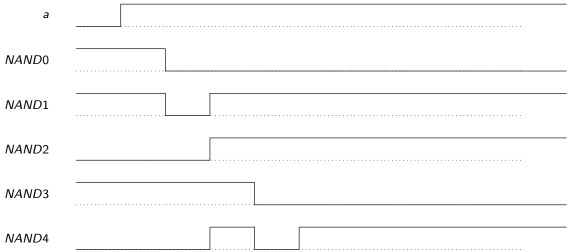
- Dynamic hazards occur in multi-level circuits in which there are $\geq 3$ paths from an input to the output and these paths have unequal delays.
- Example... assume every gate has 1 unit of delay...



$a = 0 \rightarrow 1$
$b = 1$
$c = 1$
$d = 1$
NAND0
NAND1
NAND2
NAND3
NAND4
$1 \rightarrow 1 \rightarrow 0 \rightarrow 1$
$1 \rightarrow 1 \rightarrow 0$
$0 \rightarrow 0 \rightarrow 0 \rightarrow 1$
$f$
$0 \rightarrow 0 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1$
$1 \rightarrow 1 \rightarrow 1 \rightarrow 1 \rightarrow 0$

# Illustration of dynamic hazards

- Signals drawn in time...



- Dynamic hazards are more difficult to fix and we won't consider it.
- If we wanted or needed something in which hazards were masked, we we could always implement *f* as a 2-level SOP or POS (so no dynamic hazards) and then proceed to remove static hazards.