Hardware Modeling [VU] (191.011)
– WS25 –

9-Valued Logic and Resolution (IEEE 1164)

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

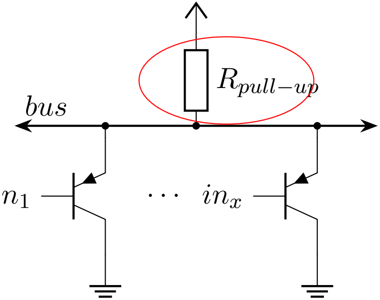
WS 2025/26

Modified: 2025-12-16, 18:02 (f8a58e9)

Motivation | Wired-AND

- Example: Wired-AND circuit
- A pull-up resistor pulls bus to HIGH when none of the transistors is active
- Setting one of the inputs $in_1, \ldots, in_x$ to LOW overrides the pull-up
  - How can we model this overriding behavior?

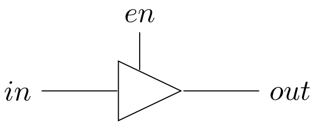
⇒ We cannot model this with Boolean values alone!



Motivation | Tri-State Buffer

- Another example: *tri-state* buffer
- $en = 1 \Rightarrow$ buffer is *transparent*: in propagated to out
- $en = 0 \Rightarrow$ buffer is *disabled*: high impedance at $out \Rightarrow$ overriding by active driver possible

  $\Rightarrow$ We cannot model this with Boolean values alone!



The IEEE std_logic_1164 package

- Recall from *Digital Design* lecture: 9-valued logic
  - Contains values for different driver strength / impedance
  - Also values useful for simulation and synthesis

  ⇒ IEEE 1164 standard for VHDL
- Implemented in the `std_logic_1164` package
- Must be imported via

```
library ieee;
use ieee.std_logic_1164.all;
```

⇒ Get access to two 9-valued logic (and related) types:

  - *Unresolved*: `std_ulogic`
  - *Resolved*: `std_logic`

IEEE 1164 Value System

The standard defines nine values and example use cases

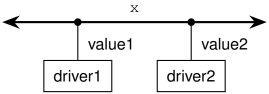
Value	Name	Example Use Case

IEEE 1164 std_ulogic Type

- Simple enumeration type with nine values

```
type std_ulogic is ('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-');
```

- *Unresolved*: Only supports signals with **single** driver
- Multiple drivers are detected and reported (during elaboration)



```
[...]: error: too many drivers for signal "x"
```

IEEE 1164 std_logic Type

- Special subtype of `std_ulogic`

```
subtype std_logic is resolved std_ulogic;
```

- `std_logic` has the same nine values as `std_ulogic`
- Allows **multiple** drivers (e.g., wired-AND, tri-state bus)
- Changing the type of signal `x` in the previous example does not result in the observed error
  - There are still multiple drivers ⇒ What value will `x` exhibit?
- Uses a *resolution function* (`resolved`)

Resolution Function

- Defines resolution of multiple drivers' values into single *resolved value*
- Single parameter: Array of all values assigned to signal
- Invoked during the simulation, no real meaning for synthesis
- Resolution functions can be associated to subtypes or signals
  - For subtypes: All signals of this subtype are resolved
  - Arrays and records of subtypes are also supported
  - For signals: Only respective signal resolved
  - Example: `signal x : resolved std_ulogic;`

The std_ulogic Resolution Function

- Resolves multiple `std_ulogic` values into a single one

The std_ulogic Resolution Function (cont'd)

- The `RES_TABLE` defines how two values are resolved into one
  Example: pull-up (`'H'`), active (`'0'`) and inactive (`'Z'`) driver

```
resolve("H0Z"):
 1  RES_TABLE('Z', 'H') = 'H'
 2  RES_TABLE('H', '0') = '0'
 3  RES_TABLE('0', 'Z') = '0'
⇒ resolve("H0Z") = '0'
```

Lecture Complete!