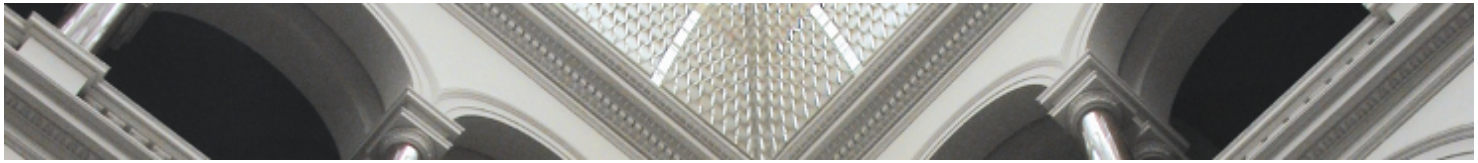
Technische Universität Berlin

# 5. Praktikumstermin

Rechnerorganisation Praktikum WiSe23/24 | Architektur eingebetteter Systeme |
Attribute, taktsynchrone Prozesse



Technische Universität Berlin

# Attribute

- Auf Attribute wird mittels Hochkomma „ ’ “ zugegriffen
- Es gibt Attribute die auf
  - Signale,

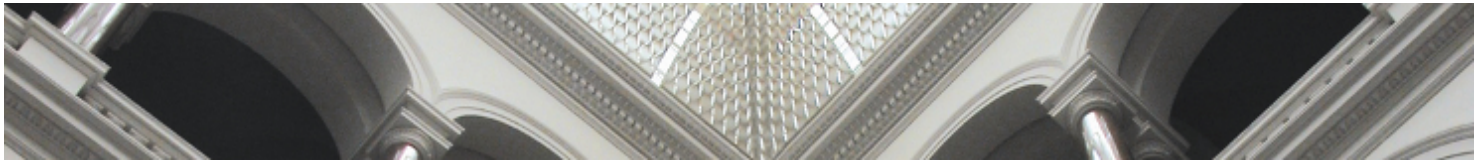
```
signal s : std_logic;
...
if s'stable then
...
```

  - Datentypen,

```
std_logic'image(s)
```

  - Arraydatentypen (nächste Folie) oder
  - Entitys

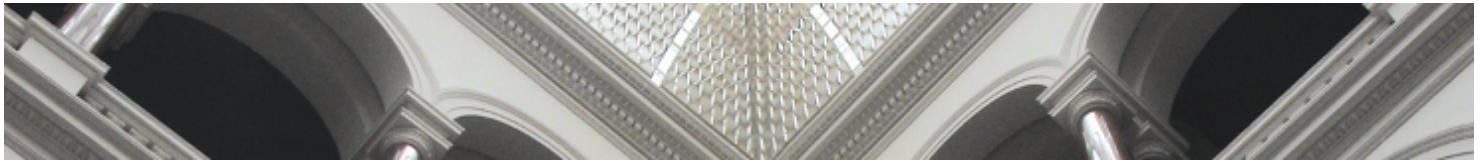
  angewandt werden können



Technische Universität Berlin

# Attribute auf `std_logic_vector`

```
architecture foo of bar is

    signal slv : std_logic_vector(3 downto 0);
    signal vec : std_logic_vector(slv'range); -- gleiche Range: 3 downto 0

begin

    process begin

        for i in slv'range loop -- 3 downto 0 oder
        ...
        for i in slv'low to slv'high loop -- 0 to 3
            vec(i) <= slv(i);
        end loop;

    end process;

    slv(slv'left) <= '0'; -- slv(3) = 0

end architecture;
```



Technische Universität Berlin

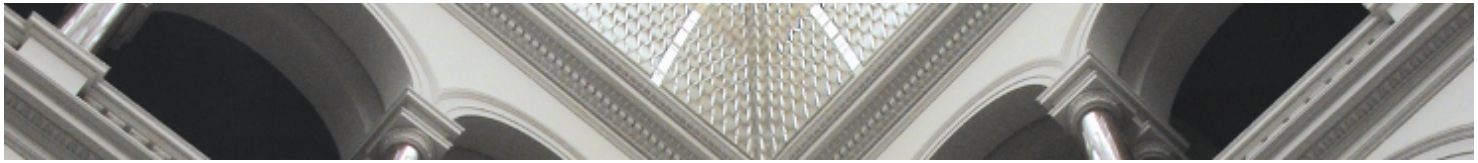
# Taktflankenerkennung mit Attributen

- Attribut `event` zeigt an, ob sich das Signal im aktuellen Simulationszyklus verändert hat
- Kombination mit Prozess mit entsprechender *sensitivity list*
- Erkennung von positiver/steigender Flanke:

```
pos_edge: process(clk)
begin
    if clk'event and clk = '1' then
        ...
    end if;
end process;
```

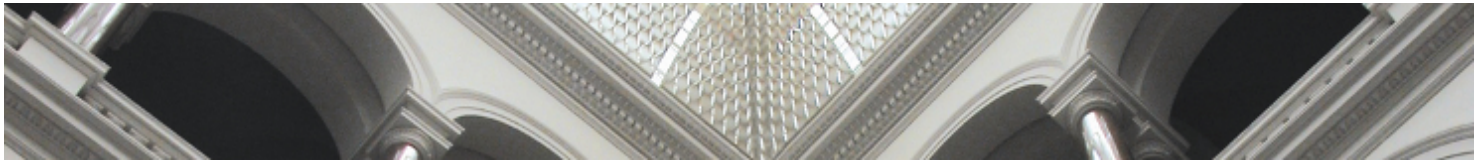
- Modernere Alternative: Funktion aus `ieee.std_logic_1164`

```
pos_edge: process(clk)
begin
    if rising_edge(clk) then
        ...
    end if;
end process;
```



# `rising_edge`: Was steckt dahinter?


```
function rising_edge (signal s : STD_ULOGIC) return BOOLEAN is
begin
  return (s'event and (To_X01(s) = '1') and
          (To_X01(s'last_value) = '0'));
end rising_edge;
```



# kombinatorische vs. taktsynchrone Prozesse

## kombinatorische Prozesse

Alle Signale, auf die innerhalb des Prozesses lesend zugegriffen wird (Signalzuweisung, `case`, `if`), sollten in der Empfindlichkeitsliste stehen!

Ausnahme: ausschließlich simulierter Code / Testbenches

```
pos_edge: process(a, b, c)
begin
    if b = '1' then
        d <= a and c;
    end if;
end process;
```

## taktsynchrone Prozesse

Nur das Taktsignal steht in der Empfindlichkeitsliste!

Ausnahme: asynchroner Reset

```
pos_edge: process(clk)
begin
    if rising_edge(clk) then
        ...
    end if;
end process;
```