FAKULTÄT FÜR INFORMATIK
TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Rechnertechnik und Rechnerorganisation
Prof. Dr. Martin Schulz

# Einführung in die Rechnerarchitektur

Wintersemester 2017/2018

Tutorübung 11: VHDL-Basiskomponenten (Lösungen) 15.01–19.01.2018

---

## VHDL Signale

1. (a)

| Signal | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| Bits | 1 | 1 | 1 | 1 | 8 | 10 | 8 | 8 | 8 |

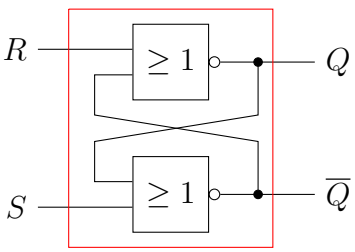
(b)
- std_logic_vector für Bitmuster, Bits können alle Werte von std_logic annehmen, neben 0 und 1 z.B. auch undefiniert (U) (weitere: X, L, H, Z, W)
- unsigned für vorzeichenlose Zahlen
- signed für vorzeichenbehaftete Zahlen
- Zuweisung, Rechnen und Vergleich mit Integerzahlen nur bei unsigned und signed möglich

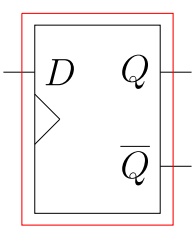
(c)
```
signal K : std_logic := '1';
signal L : std_logic_vector(3 downto 0) := "1010";
signal M : unsigned(2 downto 0) := 7;
signal N : unsigned(7 downto 0) := x"EA";
signal O : std_logic_vector(7 downto 0) := (others => 'U');
```

(d)
```
A <= not B;
B <= '1' when G = "11111111" else '0';
C <= A or (not B and D); -- Präzendenz beachten:
-- not bindet stärker, and und or binden gleich stark
D <= G(0);
E <= H(5 downto 0) & "00";
F <= ("00" &  H) + (I(7) & I(7) & I);
```

## Flip-Flop-Realisierung

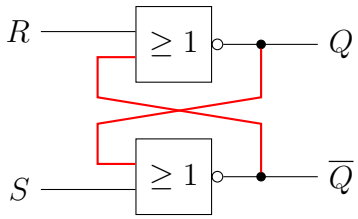
2. (a)



(b)
```
entity rsff is
    port(   s       : in  std_logic;
            r       : in  std_logic;
            q       : out std_logic;
            q_not   : out std_logic
        );
end entity;
```

(c)
```
entity dmsff is
    port(   clk     : in  std_logic;
            d       : in  std_logic;
            q       : out std_logic;
            q_not   : out std_logic
        );
end entity;
```

(d)



interne Signale: q_intern, q_not_intern, wobei

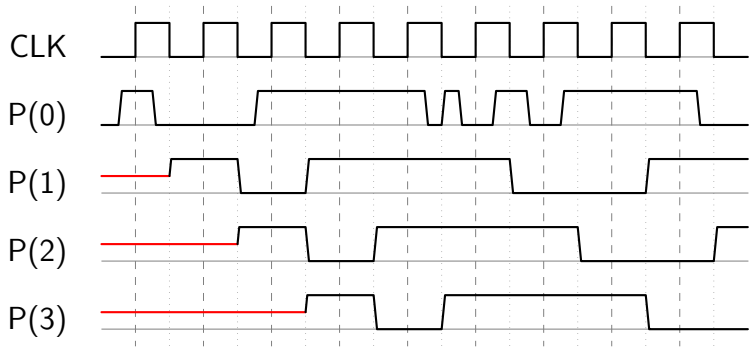
```
q_not_intern <= not q_intern;
```

(e)
```
architecture behaviour of rsff is
    signal q_intern : std_logic;
    signal q_not_intern : std_logic;
begin
    q_intern <= '1' when s = '1' else
                '0' when r = '1' else
                q_intern;

    q_not_intern <= not q_intern;

    q <= q_intern;
    q_not <= q_not_intern;
end architecture;
```

(f)
```
architecture behaviour of dmsff is
    signal q_intern : std_logic;
begin
    q <= q_intern;
    q_not <= not q_intern;

    process(clk)
    begin
        if rising_edge(clk) then
            q_intern <= d;
        end if;
    end process;
end architecture;
```

## Seriell-Parallel-Wandler

3.




4. (a)
```
entity schieberegister is
    port(   CLK : in  std_logic;
            S   : in  std_logic;
            P   : out std_logic_vector(7 downto 0)
        );
end entity;
```

(b)
```
architecture behaviour of schieberegister is
    signal p_intern : std_logic_vector(7 downto 0);
begin
    P <= p_intern;

    process(CLK)
    begin
        if rising_edge(CLK) then
            p_intern <= p_intern(6 downto 0) & S;
        end if;
    end process;
end architecture;
```

## BCD-Countdown

5. (a) $\lceil lg(10) \rceil = 4$, also 4 Bits

   (b)
```
entity COUNTDOWN is
    port(   CLK     : in  std_logic;
            VALUE   : out unsigned(3 downto 0);
            BCD     : out std_logic
        );
end entity;
```

   (c)
```
architecture v1 of COUNTDOWN is
    signal z : unsigned(3 downto 0);
begin
    BCD <= '1' when z <= 9 else '0';
    VALUE <= z;

    process(CLK)
    begin
        if rising_edge(CLK) then
            z <= z - 1;
        end if;
    end process;
end architecture;
```

6. (a)
```
entity SEVENSEGMENT is
    port(   CLK         : in  std_logic;
            VALUE       : in  unsigned(3 downto 0);
            BCD         : in  std_logic;
            SEGMENTS    : out std_logic_vector(6 downto 0)
        );
end entity;
```

   (b)
```
architecture v1 of SEVENSEGMENT is
    type rom_type is array 0 to 9 of std_logic_vector(6 downto 0);
    signal rom : rom_type;
begin
    process(CLK)
    begin
        if rising_edge(CLK) then
            if BCD = '1' then
                SEGMENTS <= rom(VALUE);
            end if;
        end if;
    end process;
end architecture;
```