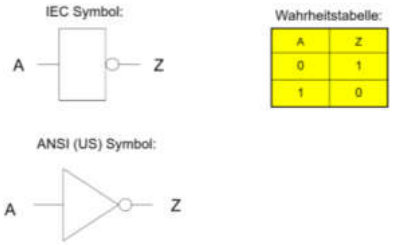
Elementare logische Verknüpfungen

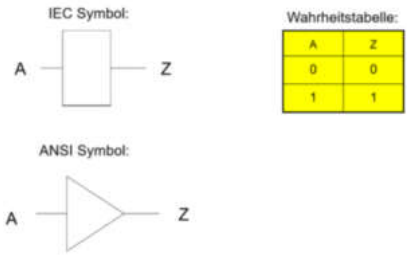
Inverter (NOT-gate)

- Invertiert das Signal Z= !A



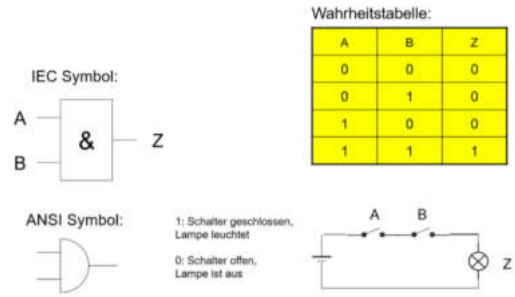
Puffer

- Verzögert das Signal



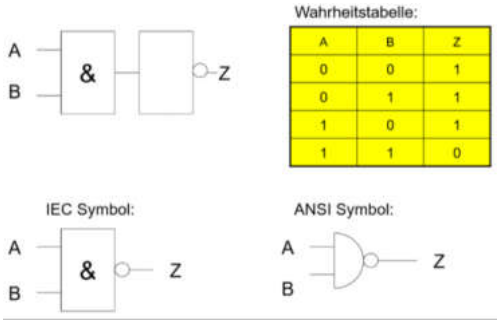
AND-gate

- Für Ausgang Z= A&B



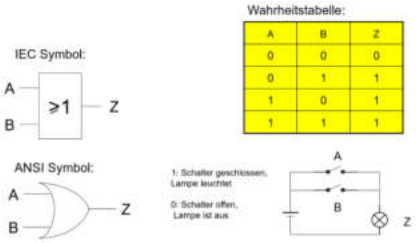
NAND-gate

- Für Ausgang Z= !(A&B)



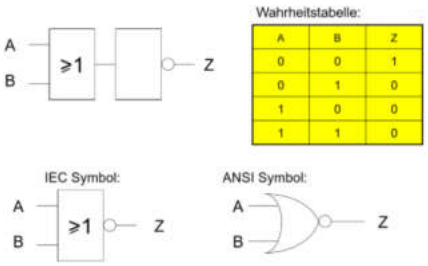
OR-gate

- Für Ausgang Z= A#B



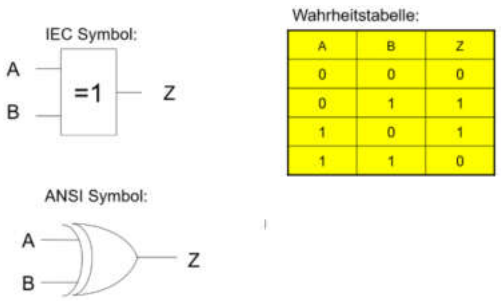
NOR-gate

- Für Ausgang Z= !(A#B)



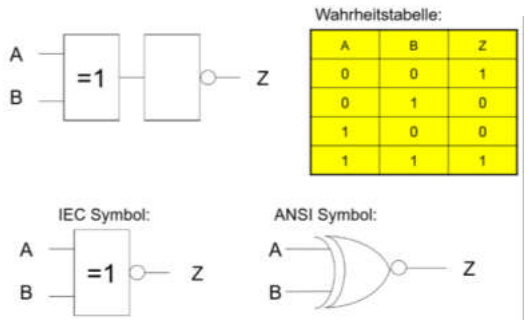
EXOR / XOR-gate

- Für Ausgang Z= A#B einzeln



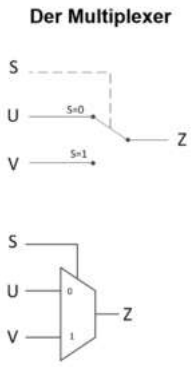
EXNOR / XNOR-gate

- Für Ausgang Z = !(A#B) einzeln



Der Multiplexer (Muxer)

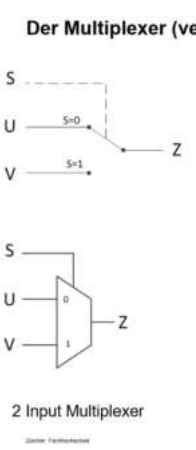
- S wählt zwischen U und V



S	U	V	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Ausführliche Wahrheitstabelle

- Muxer können vereinfacht werden



x = don't care

S	U	V	Z
0	0	x	0
0	0	x	0
0	1	x	1
0	1	x	1
1	x	0	0
1	x	1	1
1	x	0	0
1	x	1	1

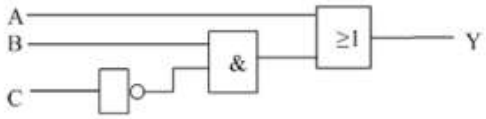
Vereinfachte Wahrheitstabelle

S	Z
0	U
1	V

Boolsche Algebra

OR-Vernüpfung: #
AND-Verknüpfung: &
NOT-Verknüpfung: !

Beispiel:



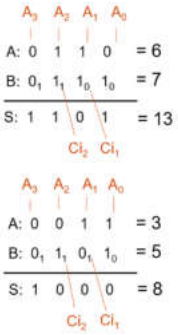
Y= A # (B & !C)

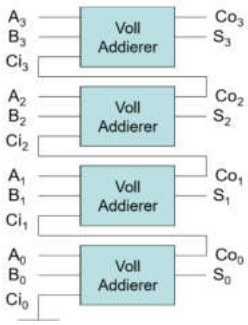
Das Binär- /Hexadezimalsystem

Dezimal	Binär	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	b
12	1100	C
13	1101	d
14	1110	E
15	1111	F

Binärarithmetik

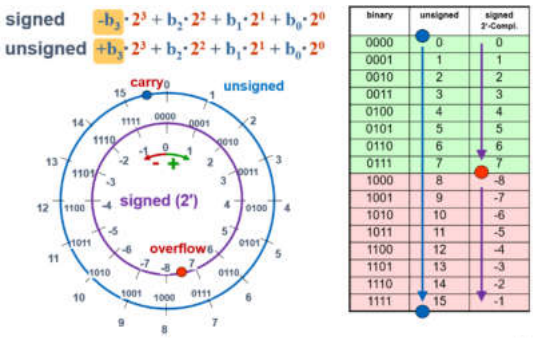
-





Zweierkomplementzahlen (signed/ unsigned)

- Negative Zahlen können dargestellt werden -> Achtung Over/Underflow!



Beispiel 4-Bit:
Von 0 bis 7 und -8 bis -1

2-er Potenz-reihe

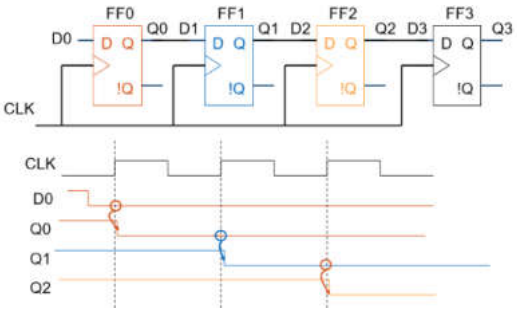
Potenz	Resultat
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16'384
15	32'768
16	65'536

VHDL: Programmaufbau

- Eingangslogik
- Programm
- Ausgangslogik

VHDL: Schieberegister

- Besteht aus Flip-Flops



Beispiel: Schieberegister

Shift to MSB

```
next_shiftreg(3 downto 0) <= shiftreg(2 downto 0) & serial_in;
   (3)..(2)..(1)..(0)     <=  (2)..(1)..(0).. serial_in
```

Shift to LSB

```
next_shiftreg(3 downto 0) <= serial_in & shiftreg(3 downto 1);
   (3)..(2)..(1)..(0)  <=  serial_in ..(3)..(2)..(1)
```

In code:

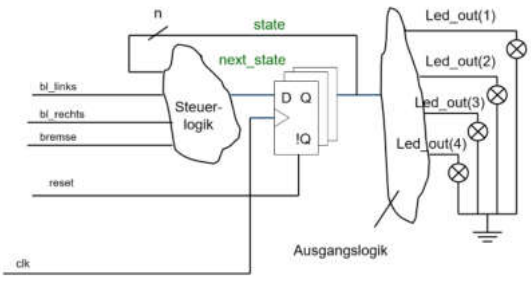
```
ARCHITECTURE rtl OF schieberegister IS
    SIGNAL shiftreg     :       std_logic_vector(3 downto 0);
    SIGNAL next_shiftreg  :     std_logic_vector(3 downto 0);
BEGIN
    logik : PROCESS (all)                                   Logik-Prozess
    BEGIN
        IF shift_to_lsb = '1' THEN
            next_shiftreg <= serial_in & shiftreg(3 downto 1);
        ELSE
            next_shiftreg <= shiftreg(2 downto 0) & serial_in;
    END IF;
    END PROCESS logik;

    flip_flops : PROCESS (all)                              Taktprozess
    BEGIN
        IF reset_n = '0' THEN
            shiftreg <= (OTHERS => '0');
        ELSIF rising_edge(clk) THEN
            shiftreg <= next_shiftreg;
        END IF;
    END PROCESS flip_flops;

    serial_out <= shiftreg(3);
END ARCHITECTURE rtl;
```

Automaten – FSM (Finite State Machine)

- Logikaufbau: Beispiel: Knight rider



Automaten: Bubble (RTL)-Diagramm

- Ermitteln der Zustände
- Ermitteln der Zustandsübergänge
- Zeichnen