```
                          Beispiele_Decoder_Multiplexer
%       AHDL Entwurfsbeispiele für Dekoder und Multiplexer
        und Demultiplexer in verschieden Entwurfsstilen.
        Die Beispiele sind ohne Kommentare angegeben.
        Dadurch soll die Beschäftigung mit den Beispielen
        angeregt werden
%


TITLE "1 aus 4 Dekoder mit case";

SUBDESIGN decoder
(
        code[1..0]      : INPUT;
        out[3..0]               : OUTPUT;
)
BEGIN
        CASE code[] IS
                WHEN 0 => out[] = B"0001";
                WHEN 1 => out[] = B"0010";
                WHEN 2 => out[] = B"0100";
                WHEN 3 => out[] = B"1000";
        END CASE;
END;

_____________________________________________________________________________
__


TITLE "Adressdekoder für eine Adresse";

SUBDESIGN decode1
(
        address[15..0]          : INPUT;
        chip_enable                     : OUTPUT;
)
BEGIN
        chip_enable = (address[15..0] == H"0370");
END;

_____________________________________________________________________________
__

TITLE "Adressdekoder für einen Adressbereich";

SUBDESIGN decoden
(
        address[15..0]          : INPUT;
        chip_enable                     : OUTPUT;
)
BEGIN
        chip_enable = (address[15..0] >= H"0370" and address[15..0] <= H"0385");
END;

_____________________________________________________________________________
__

TITLE "Dekoder mit 2 Ausgängen für verschiedene Adressbereiche";

CONSTANT ioadr   = H"0370";
CONSTANT ramadrl = H"0380";
CONSTANT ramadrh = H"038F";

SUBDESIGN decode2
(
        a[15..0]        : INPUT;
        ceio,ceram      : OUTPUT;

)
```

```
BEGIN
        ceio  = (a[15..0] == ioadr);
        ceram = (a[15..0] >= ramadrl and a[15..0] <= ramadrh);
END;
```

```
TITLE "Adressdekoder für verschiedene Adressbereiche mit Tabelle"

SUBDESIGN decode3
(
        addr[15..0], m/io                       : INPUT;
        rom, ram, print, sp[2..1]       : OUTPUT;
)
BEGIN
        TABLE
                m/io,   addr[15..0]                 => rom, ram,        print,
sp[];
                1,              B"00XXXXXXXXXXXXXX" =>  1,   0, 0,     B"00";
                1,              B"100XXXXXXXXXXXXX" =>  0,   1, 0,     B"00";
                0,              B"0000001010101110" =>  0,   0, 1,     B"00";
                0,              B"0000001011011110" =>  0,   0, 0,     B"01";
                0,              B"0000001101110000" =>  0,   0, 0,     B"10";
        END TABLE;
END;
```

```
TITLE " 1 aus 4 Dekoder mit Eingang enable";

SUBDESIGN dekoder_1aus4
(
        adr[1..0],enable        : INPUT;
        dekaus[3..0]            : OUTPUT;
)

BEGIN
        dekaus0 =enable & !adr1 & !adr0;
        dekaus1 =enable & !adr1 & adr0;
        dekaus2 =enable & adr1 & !adr0;
        dekaus3 =enable & adr1 & adr0;
END;
```

```
TITLE " 1 aus 4 Dekoder mit Eingang enable";

SUBDESIGN dekoder_1aus4_wt
(
        adr[1..0],enable        : INPUT;
        dekaus[3..0]            : OUTPUT;
)
% Funktionsbeschreibung mit Wahrheitstabelle %
BEGIN
TABLE
        adr[],  enable          =>      dekaus[];
        B"XX",  0                       =>      B"0000";
        B"00",  1                       =>      B"0001";
        B"01",  1                       =>      B"0010";
        B"10",  1                       =>      B"0100";
```

```
        B"11",  1                       =>      B"1000";
END TABLE;
END;
```

```
TITLE " 1 aus 4 Dekoder mit Eingang enable";

SUBDESIGN dekoder_1aus4_case
(
        adr[1..0],enable        : INPUT;
        dekaus[3..0]            : OUTPUT;
)
% Funktionsbeschreibung mit Wahrheitstabelle %
BEGIN

CASE (enable,adr[]) IS  % Gruppenbildung %
        WHEN B"100" =>
                dekaus[]=B"0001";
        WHEN B"101" =>
                dekaus[]=B"0010";
        WHEN B"110" =>
                dekaus[]=B"0100";
        WHEN B"111" =>
                dekaus[]=B"1000";
        WHEN OTHERS =>
                dekaus[]=B"0000";
END CASE;
END;
```

```
TITLE " 7 Segmentdekoder Segmente leuchten bei High-Pegel"

SUBDESIGN Dekoder_7Segment

%   -a-                                                         %
% f|   |b                                                       %
%   -g-                                                         %
% e|   |c                                                       %
%   -d-                                                         %
%                                                                       %
%  0 1 2 3 4 5 6 7 8 9 A b C d E F      %
(
        i[3..0]                                 : INPUT;
        a, b, c, d, e, f, g                             : OUTPUT;
)
BEGIN
        TABLE
                i[3..0]         =>      a, b, c, d, e, f, g;

                H"0"            =>      1, 1, 1, 1, 1, 1, 0;
                H"1"            =>      0, 1, 1, 0, 0, 0, 0;
                H"2"            =>      1, 1, 0, 1, 1, 0, 1;
                H"3"            =>      1, 1, 1, 1, 0, 0, 1;
                H"4"            =>      0, 1, 1, 0, 0, 1, 1;
                H"5"            =>      1, 0, 1, 1, 0, 1, 1;
                H"6"            =>      1, 0, 1, 1, 1, 1, 1;
                H"7"            =>      1, 1, 1, 0, 0, 0, 0;
                H"8"            =>      1, 1, 1, 1, 1, 1, 1;
                H"9"            =>      1, 1, 1, 1, 0, 1, 1;
                H"A"            =>      1, 1, 1, 0, 1, 1, 1;
                H"B"            =>      0, 0, 1, 1, 1, 1, 1;
```

```
                H"C"            =>      1, 0, 0, 1, 1, 1, 0;
                H"D"            =>      0, 1, 1, 1, 1, 0, 1;
                H"E"            =>      1, 0, 0, 1, 1, 1, 1;
                H"F"            =>      1, 0, 0, 0, 1, 1, 1;
        END TABLE;
END;
```

```
TITLE "Dekoder BCD zu 7 Segment Segmente leuchten bei Low-Pegel";

SUBDESIGN dekoder7seg
(
        bcd[3..0]       : INPUT;
        hex[6..0]       : OUTPUT;
)
BEGIN
TABLE
        bcd[]           =>      hex[];
        B"0000"         =>      B"1000000";
        B"0001"         =>      B"1111001";
        B"0010"         =>      B"0100100";
        B"0011"         =>      B"0110000";
        B"0100"         =>      B"0011001";
        B"0101"         =>      B"0010010";
        B"0110"         =>      B"0000010";
        B"0111"         =>      B"1111000";
        B"1000"         =>      B"0000000";
        B"1001"         =>      B"0010000";
        B"1010"         =>      B"0001000";
        B"1011"         =>      B"0000011";
        B"1100"         =>      B"1000110";
        B"1101"         =>      B"0100001";
        B"1110"         =>      B"0000110";
        B"1111"         =>      B"0001110";
END TABLE;

END;
```

```
TITLE " 4 zu 1 Demultiplexer"

SUBDESIGN demux_1aus4
(
        DIN, ADR[1..0]  : INPUT;
        DOUT[3..0]              : OUTPUT;
)
BEGIN
DEFAULTS
        DOUT[]=B"0000";
END DEFAULTS;

TABLE
        ADR[],  DIN     =>      DOUT[];
        0,              1       =>      B"0001";
        1,              1       =>      B"0010";
        2,              1       =>      B"0100";
        3,              1       =>      B"1000";

END TABLE;
```

```
END;
```

```
%       Funktionsbeschreibung für einen
        4 zu 1 Multiplexer
        mit Logikgleichungen %

SUBDESIGN mux_log
(
        di[3..0]                : INPUT;        --Datenvektor
        adr[1..0]               : INPUT;        --Adressvektor
        muxaus                  : OUTPUT;
)
BEGIN
        muxaus= di[0]&!adr[1]&!adr[0]#
                        di[1]&!adr[1]&adr[0]#
                        di[2]&adr[1]&!adr[0]#
                        di[3]& adr[1]& adr[0];
END;
```

```
%       Funktionsbeschreibung für einen
        4 zu 1 Multiplexer
        mit Wahheitstabelle %
SUBDESIGN mux_wt
(
        di[3..0]                : INPUT;        --Datenvektor
        adr[1..0]               : INPUT;        --Adressvektor
        muxaus                  : OUTPUT;
)
BEGIN
        TABLE
        di[],   adr[]           => muxaus;
        B"XXX0",B"00"           => 0;
        B"XXX1",B"00"           => 1;
        B"XX0X",B"01"           => 0;
        B"XX1X",B"01"           => 1;
        B"X0XX",B"10"           => 0;
        B"X1XX",B"10"           => 1;
        B"0XXX",B"11"           => 0;
        B"1XXX",B"11"           => 1;
END TABLE;

END;
```

```
%       Funktionsbeschreibung für einen
        4 zu 1 Multiplexer
        mit bedingter Signalzuweisung
        mit CASE
%
SUBDESIGN mux_case
(
        di[3..0]                : INPUT;        --Datenvektor
        adr[1..0]               : INPUT;        --Adressvektor
        muxaus                  : OUTPUT;
)
BEGIN
```

```
CASE adr[] IS
        WHEN 0 =>
                muxaus=di[0];
        WHEN 1 =>
                muxaus=di[1];
        WHEN 2 =>
                muxaus=di[2];
        WHEN OTHERS =>
                muxaus=di[3];
END CASE;
END;
```

```
TITLE "n-fach 2zu 1 Multiplexer";

CONSTANT wortbreite = 8;

SUBDESIGN mux_nfach_2zu1
(
        sel                                                     : INPUT ;
        dina[wortbreite-1.. 0]  : INPUT;
        dinb[wortbreite-1.. 0]  : INPUT;
        daus[wortbreite-1.. 0]  : OUTPUT;
)

BEGIN
        IF sel THEN
                daus[] = dina[];
        ELSE
                daus[] = dinb[];
        END IF;
END;
```

```
TITLE "n-fach 2zu1 parametrisierbarer Multiplexer";

PARAMETERS
(
  wortbreite=8
);

SUBDESIGN mux_nfach_2zu1_parameter
(
        sel                                                     : INPUT ;
        dina[wortbreite-1.. 0]  : INPUT;
        dinb[wortbreite-1.. 0]  : INPUT;
        daus[wortbreite-1.. 0]  : OUTPUT;
)

BEGIN
        IF sel THEN
                daus[] = dina[];
        ELSE
                daus[] = dinb[];
        END IF;
END;
```

```
TITLE " 8-fach 4 zu 1 Multiplexer für Datenwege"

SUBDESIGN mux8x4_1
(
        din[3..0][7..0]         : INPUT;
        adr[1..0]                       : INPUT;
        dout[7..0]                      : OUTPUT;
)
BEGIN
CASE adr[] IS
        WHEN 0 =>
                dout[]=din[0][];
        WHEN 1 =>
                dout[]=din[1][];
        WHEN 2 =>
                dout[]=din[2][];
        WHEN 3 =>
                dout[]=din[3][];
END CASE;

END;
```