Protokoll zum Praktikum Programmierbare Schaltkreise

Christian Kroh
Matrikelnummer: 3755154
Studiengang: Informatik (Diplom)
Jahrgang: 2010/2011

15. November 2013, Dresden

Aufgabe 1 - Binär-Dekoder

1.1 Entwurf

Input 4-Bit Binärzahl durch Schieberegister SW3 ... SW0

Output 7-Segmente Darstellung einer Hexadezimalziffer (8 Einzelsignale = 7 Segmente + 1 Punkt)

Input				Output								
SW3	SW2	SW1	SW0	Hex	A	B	C	D	E	F	G	dot
0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	1	1	1	0	0	1	1	1	1	1
0	0	1	0	2	0	0	1	0	0	1	0	1
0	0	1	1	3	0	0	0	0	1	1	0	1
0	1	0	0	4	1	0	0	1	1	0	0	1
0	1	0	1	5	0	1	0	0	1	0	0	1
0	1	1	0	6	0	1	0	0	0	0	0	1
0	1	1	1	7	0	0	0	1	1	1	1	1
1	0	0	0	8	0	0	0	0	0	0	0	1
1	0	0	1	9	0	0	0	0	1	0	0	1
1	0	1	0	A	0	0	0	1	0	0	0	1
1	0	1	1	b	1	1	0	0	0	0	0	1
1	1	0	0	C	0	1	1	0	0	0	1	1
1	1	0	1	d	1	0	0	0	0	1	0	1
1	1	1	0	E	0	1	1	0	0	0	0	1
1	1	1	1	F	0	1	1	1	0	0	0	1

1.2 Auswertung

Ressourcenbedarf 7 Logik-Elemente und 12 Pins

Aufgabe 2 - Hamming-Distanz

2.1 Entwurf

Input 2 4-Bit Werte

- 1.Wert: 4-Bit Binärzahl durch Schieberegister SW3 ... SW0
- 2.Wert: 4-Bit Binärzahl durch Schieberegister SW7 ... SW4

Output 7-Segmente Darstellung einer Hexadezimalziffer (8 Einzelsignale = 7 Segmente + 1 Punkt)

Ansatz SW3 ... SW0 und SW7 ... SW4 logisch xor verknüpfen und Ergebnis direkt auf 7-Segmente Anzeige mappen

2.2 Auswertung

Ressourcenbedarf 9 Logik-Elemente und 16 Pins

Aufgabe 3 - Modulo-n-Zähler

3.1 Entwurf

a) Der Zähler ist nach 50 Millionen Schritten zurückzusetzen (50MHz Takt entspricht 50 Millionen Taktperioden pro Sekunde)

b) Für das Schieberegister ist der Zählerzustand ein Enable-Signal

c) **Input**
- 50MHz Takt
- Reset (Schiebeschalter SW0)

Output LED-Zeile

Ansatz 2 Komponenten Schieber und Zähler

Aufgabe 1 - Binär-Dekoder

4.1 Entwurf

Anhang

5.1 01-Aufgabe Code

Listing 1: VHDL-Code Decoder

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity Decoder is
6
7    port (
8      sw    : in  std_logic_vector(3 downto 0);
9      cc    : out std_logic_vector(7 downto 0));
10
11 end Decoder;
12
13 architecture dec1 of Decoder is
14 begin
15
16   ------------------------------------------
17   -- Outputs
18   ------------------------------------------
19   with sw select
20     cc <= "00000011" when "0000",
21           "10011111" when "0001",
22           "00100101" when "0010",
23           "00001101" when "0011",
24           "10011001" when "0100",
25           "01001001" when "0101",
26           "01000001" when "0110",
27           "00011111" when "0111",
28           "00000001" when "1000",
29           "00001001" when "1001",
30           "00010001" when "1010",
31           "11000001" when "1011",
32           "01100011" when "1100",
33           "10000101" when "1101",
34           "01100001" when "1110",
35           "01110001" when "1111";
36 end dec1;
```

5.2 02-Aufgabe Code

Listing 2: VHDL-Code Hamming-Distanz

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity Hamming is
6
7    port (
8      sw1    : in  std_logic_vector(3 downto 0);
9      sw2    : in  std_logic_vector(3 downto 0);
10     cc     : out std_logic_vector(7 downto 0));
11
12 end Hamming;
13
14 architecture ham1 of Hamming is
15   signal xo : std_logic_vector(3 downto 0);
16 begin
17
18   xo <= sw1 xor sw2;
```

```
19
20    with xo select
21      cc <= "00000011" when "0000",
22            "10011111" when "0001",
23            "10011111" when "0010",
24            "00100101" when "0011",
25            "10011111" when "0100",
26            "00100101" when "0101",
27            "00100101" when "0110",
28            "00001101" when "0111",
29            "10011111" when "1000",
30            "00100101" when "1001",
31            "00100101" when "1010",
32            "00001101" when "1011",
33            "00100101" when "1100",
34            "00001101" when "1101",
35            "00001101" when "1110",
36            "10011001" when "1111";
37  end ham1;
```