Spezifikation zum Projekt Nr. 1 der Gruppe 7 (ss11-g07)

Tutor: Steffen Bauereiss

Kurzbeschreibung

Es ist ein System zu realisieren, das eine Binärzahl mit 17 Stellen in eine BCD-Zahl mit 5 x 4 Stellen konvertiert. Der Definitionsbereich liegt zwischen 0 und 99999. Die VHDL-Entity des Systems folgt.

```
entity BINBCD is

port(
    clk : in std_logic;
    bin_input : in std_logic_vector (16 downto 0);
    einer, zehner, hunderter, tausender, zehntausender : out std_logic_vector (3 downto
0);
    overflow : out std_logic
);

end BINBCD;
```

Alternative 1: Double-Dabble

Diese Alternative sieht es vor, dass die Eingabe zuerst auf ihre Gültigkeit (Eingabe <= 99999) überprüft wird. Falls die Eingabe im Definitionsbereich liegt, so wird ein zusammengesetzter 37-Bit-Vektor erstellt, der wie folgt aufgebaut ist (beginnend mit dem höchstwertigsten Bit, MSB):

```
4-Bit BCD Zehntausender, 4-Bit BCD Tausender, 4-Bit BCD Hunderter, 4-Bit BCD Zehner,
4-Bit BCD Einer, 17-Bit binäre Eingabe.
```

Danach wird nach folgendem Schema vorgegangen, um die binäre Eingabe (in den 17 niedigstwertigen Bits des zusammengesetzten Vektors) in eine BCD umzuwandeln:

1. Schiebe den Vektor um ein Bit nach links*.

2. Wenn der Binäre Wert in einer der BCD Spalten größer oder gleich 5 ist, addiere den Wert 3 zu dieser Spalte.
3. Wiederhole dieses Schema 16 mal ab Punkt 1.

*Nach 17-maligem Schieben wird abgebrochen und die Zahl liegt im BCD Format an den (4-Bit-breiten) Stellen der Zehntausender, Tausender, Hunderter, Zehner und Einer.

Falls die Eingabe nicht im Definitionsbereich liegt, werden die Ausgänge auf 0 und das Overflowbit auf 1 gesetzt.

Alternative 2: Bin-Dec-BCD

Diese Alternative sieht es vor, dass die Eingabe zuerst (mit Hilfe einer binären Konstanten: $1100001101001111 1_2 = 99999_{10}$) auf ihre Gültigkeit überprüft wird. Liegt die Eingabe im Definitionsbereich, wird sie nach folgendem Schema in eine dezimale Zahl umgewandelt (beginnend mit dem niedrigstwertigen Bit, LSB):

1. Initialisiere eine Variable "dec" mit 0. Diese wird am Ende die dezimale Zahl enthalten.
2. Multipliziere die Wertigkeit der aktuellen Stelle (2^0, 2^1, ..., 2^{16}) mit der Ziffer der aktuellen Stelle (1,0).
3. Addiere das Produkt aus Schritt 2 zu dec.
4. Wiederhole dieses Schema 16 mal ab Punkt 2 (für alle Stellen der Binären Eingabe, insg. 17 mal).

Nun, da die Eingabe in dezimaler Form vorliegt, wird jede der (maximal fünf) Stellen mit Hilfe der untenstehenden Lookup-Tabelle in eine 4-bit-breite Binärzähl umgewandelt und an den entsprechenden Ausgang gelegt.

Decimal:	0	1	2	3	4	5	6	7	8	9
Binär :	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Falls die Eingabe nicht im Definitionsbereich liegt, wird der Ausgang auf 0 und das Overflowbit auf 1 gesetzt.

Bewertung der Alternativen

Bei der Alternative 1 (A1) werden nur Schiebe- und Additionsoperationen benötigt, wogegen Alternative 2 (A2) Additions- und

Multiplikationsoperationen benötigt. Die Multiplikation würde die Hardware unnötig komplex machen.

Weiterhin wird die Eingabe bei A2 zwei mal in ein anderes Zahlensystem konvertiert, was unnötige Rechenschritte darstellt. Bei A1 kann die binäre Eingabe innerhalb einer Schleife, welche über den Vektor läuft, direkt berechnet werden.

Der Vorteil von A1 ist ihre Effizienz.

Der Vorteil von A2 ist ihre Anschaulichkeit.

Implementierungsentscheidung

Wir haben uns für den Double-Dabble-Algorithmus aus Alternative 1 entschieden, da die oben genannten Vorteile der Alternative 1 über Alternative 2 deutlich überwiegen.