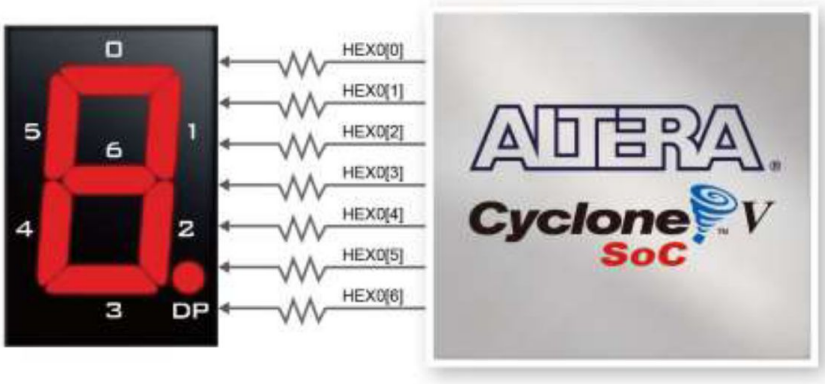
# Example 1 - 7-segment displays

The DE1-SoC board includes 6 7-segment displays connected to the FPGA.

## Overview

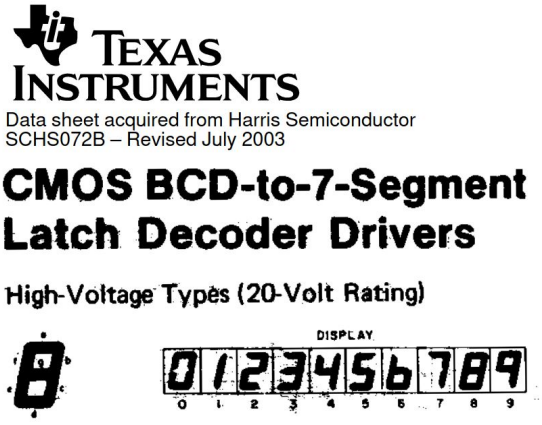
Each of the segments is connected to a different pin of the FPGA. The displays are common anode, which means that the positive connection is common for every LED in the display and the display is controlled by the cathodes, which activate the segments when pulled low.




## Driving 7-segment displays

A (once) common convention for storing decimal numbers in a binary digital system is BCD - binary-coded decimal. This encoding stores a single digit in 4 bits, with the first 10 codes representing digits 0 to 9. The remaining codes are unused. This way, a single byte can encode the numbers between 0 and 99. Many computer architectures, including x86, include instructions which help operate on BCD numbers.

The BCD digit needs to be decoded into the segment signals for display. A dedicated integrated circuit were used for this, such as the 4511:



## Double dabble

Decoding binary numbers into decimal is rather tricky. The usual repeated division by 10 and taking the remainder is not very practical on a FPGA, as dividers are expensive. At the cost of latency (which is not a consideration for driving human-readable displays), the algorithm known as double-dabble converts binary numbers into BCD.

The algorithm converts a *n* digit binary number into *d* digit BCD-coded decimal number. For example, an 8-bit number into 3 decimal digits. A scratch space is reserved for *d*4* + *n* bit values, and the low *n* bits are set to the input binary number. For example, converting 8'd179 = 8'b10110011 to BCD:

```
0000 0000 0000 10110011
```

The conversion process then takes *n* iterations. In each iteration, each of the *d* BCD digits are checked if they are larger than 4. If true, 3 is added to the digit. The whole register is then shifted by 1 position to the left:

```
0000 0000 0000 10110011
0000 0000 0001 01100110 shift
0000 0000 0010 11001100 shift
0000 0000 0101 10011000 shift
0000 0000 1000 10011000 add 3 to BCD digit 0
0000 0001 0001 00110000 shift
0000 0010 0010 01100000 shift
0000 0100 0100 11000000 shift
0000 1000 1001 10000000 shift
0000 1000 1100 10000000 add 3 to BCD digit 0
0000 1011 1100 10000000 add 3 to BCD digit 1
0001 0111 1001 00000000 shift
```

The result is then available in the top *d*4* bits of the register.

## Example design

The design contains a free-running counter. The high bits of this counter drive a combinatorial double-dabble module. The module converts the binary number into six decimal digits, encoded as BCD.

The BCD digits are then connected to BCD-to-7segment decoders which in turn drive the displays.