# Binary-to-BCD Converter

# Shift and Add-3 Algorithm

1. Shift the binary number left one bit.
2. If 8 shifts have taken place, the BCD number is in the *Hundreds*, *Tens*, and *Units* column.
3. If the binary value in any of the BCD columns is 5 or greater, add 3 to that value in that BCD column.
4. Go to 1.

| Operation | Hundreds | Tens | Units | Binary | |
|---|---|---|---|---|---|
| HEX | | | | F | F |
| Start | | | | 1 1 1 1 | 1 1 1 1 |

# Steps to convert an 8-bit binary number to BCD

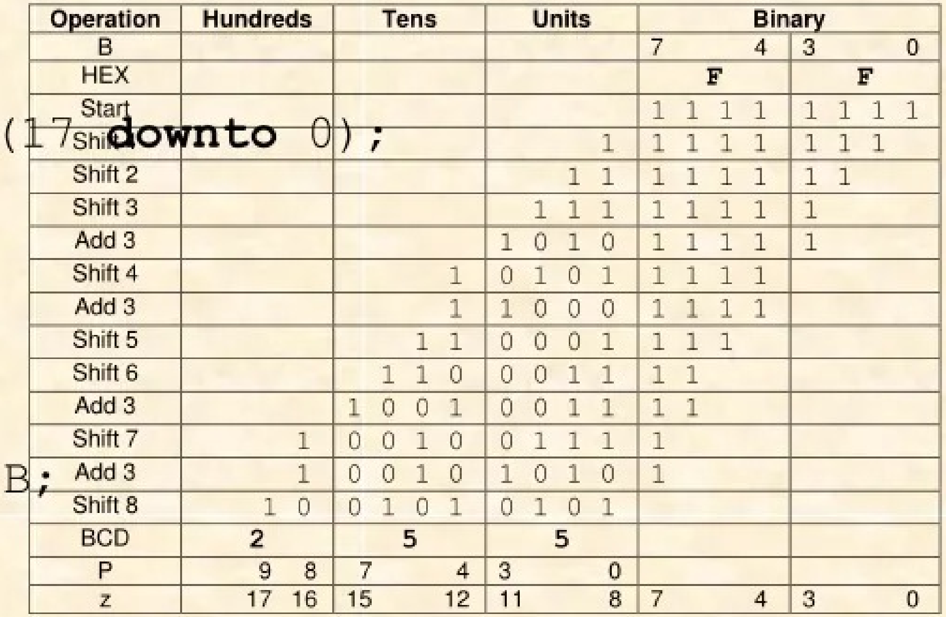
| Operation | Hundreds | Tens | Units | Binary | |
|---|---|---|---|---|---|
| B | | | | 7 4 | 3 0 |
| HEX | | | | **F** | **F** |
| Start | | | | 1 1 1 1 | 1 1 1 1 |
| Shift 1 | | | 1 | 1 1 1 1 | 1 1 1 |
| Shift 2 | | | 1 1 | 1 1 1 1 | 1 1 |
| Shift 3 | | | 1 1 1 | 1 1 1 1 | 1 |
| Add 3 | | | 1 0 1 0 | 1 1 1 1 | 1 |
| Shift 4 | | 1 | 0 1 0 1 | 1 1 1 1 | |
| Add 3 | | 1 | 1 0 0 0 | 1 1 1 1 | |
| Shift 5 | | 1 1 | 0 0 0 1 | 1 1 1 | |
| Shift 6 | | 1 1 0 | 0 0 1 1 | 1 1 | |
| Add 3 | | 1 0 0 1 | 0 0 1 1 | 1 1 | |
| Shift 7 | 1 | 0 0 1 0 | 0 1 1 1 | 1 | |
| Add 3 | 1 | 0 0 1 0 | 1 0 1 0 | 1 | |
| Shift 8 | 1 0 | 0 1 0 1 | 0 1 0 1 | | |
| BCD | **2** | **5** | **5** | | |
| P | 9 8 | 7 4 | 3 0 | | |
| z | 17 16 | 15 12 | 11 8 | 7 4 | 3 0 |

# Example of converting hex E to BCD

| Operation | Tens | Units | Binary |
|---|---|---|---|
| HEX | | | E |
| Start | | | 1 1 1 0 |
| Shift 1 | | 1 | 1 1 0 |
| Shift 2 | | 1 1 | 1 0 |
| Shift 3 | | 1 1 1 | 0 |
| Shift 4 | | 1 1 1 0 | |
| 6 | | 0 1 1 0 | |
| Add 6 | 1 | 0 1 0 0 | |
| BCD | 1 | 4 | |

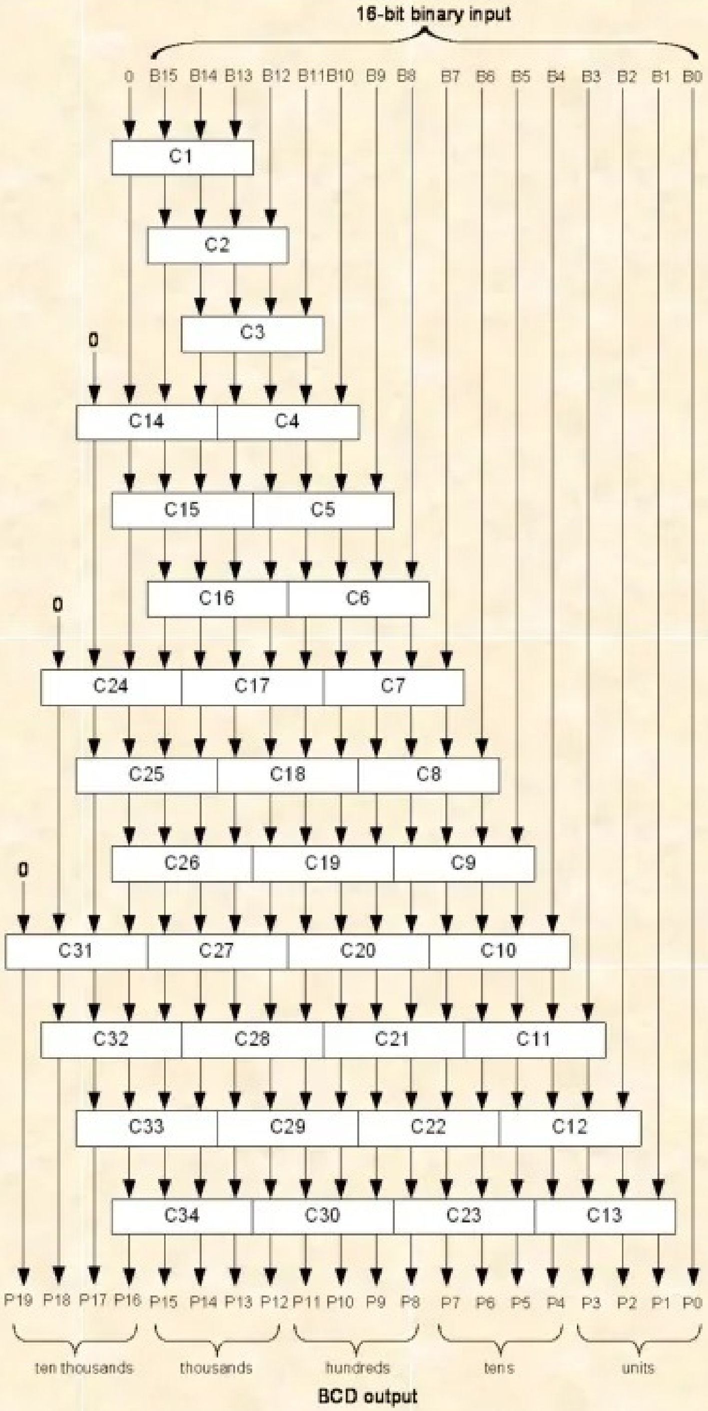
```
-- Title: Binary-to-BCD Converter
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity binbcd is
    port (
        B: in STD_LOGIC_VECTOR (7 downto 0);
        P: out STD_LOGIC_VECTOR (9 downto 0)
    );
end binbcd;
```

```
architecture binbcd_arch of binbcd is
begin
  bcd1: process(B)

  variable z: STD_LOGIC_VECTOR (17 downto 0);

  begin
      for i in 0 to 17 loop
            z(i) := '0';
      end loop;
            z(10 downto 3) := B;

      for i in 0 to 4 loop
            if z(11 downto 8) > 4 then
                  z(11 downto 8) := z(11 downto 8) + 3;
            end if;
            if z(15 downto 12) > 4 then
                  z(15 downto 12) := z(15 downto 12) + 3;
            end if;
            z(17 downto 1) := z(16 downto 0);
      end loop;
      P <= z(17 downto 8);
  end process bcd1;
end binbcd_arch;
```



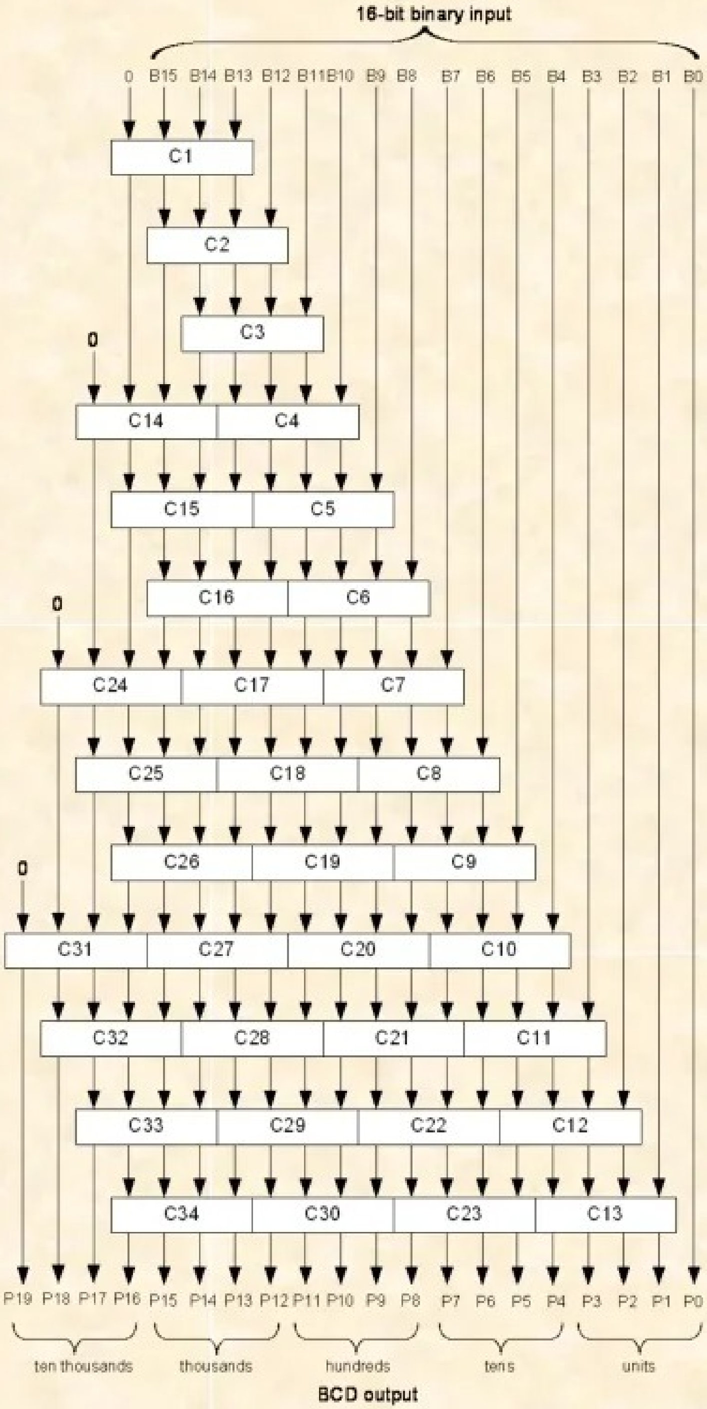
| Operation | Hundreds | Tens | Units | Binary | |
|---|---|---|---|---|---|
| B | | | | 7 4 | 3 0 |
| HEX | | | | **F** | **F** |
| Start | | | | 1 1 1 1 | 1 1 1 1 |
| Shift 1 | | | 1 | 1 1 1 1 | 1 1 1 |
| Shift 2 | | | 1 1 | 1 1 1 1 | 1 1 |
| Shift 3 | | | 1 1 1 | 1 1 1 1 | 1 |
| Add 3 | | | 1 0 1 0 | 1 1 1 1 | 1 |
| Shift 4 | | 1 | 0 1 0 1 | 1 1 1 1 | |
| Add 3 | | 1 | 1 0 0 0 | 1 1 1 1 | |
| Shift 5 | | 1 1 | 0 0 0 1 | 1 1 1 | |
| Shift 6 | | 1 1 0 | 0 0 1 1 | 1 1 | |
| Add 3 | | 1 0 0 1 | 0 0 1 1 | 1 1 | |
| Shift 7 | 1 | 0 0 1 0 | 0 1 1 1 | 1 | |
| Add 3 | 1 | 0 0 1 0 | 1 0 1 0 | 1 | |
| Shift 8 | 1 0 | 0 1 0 1 | 0 1 0 1 | | |
| BCD | **2** | **5** | **5** | | |
| P | 9 8 | 7 4 | 3 0 | | |
| z | 17 16 | 15 12 | 11 8 | 7 4 | 3 0 |

# 16-bit Binary-to-BCD Converter

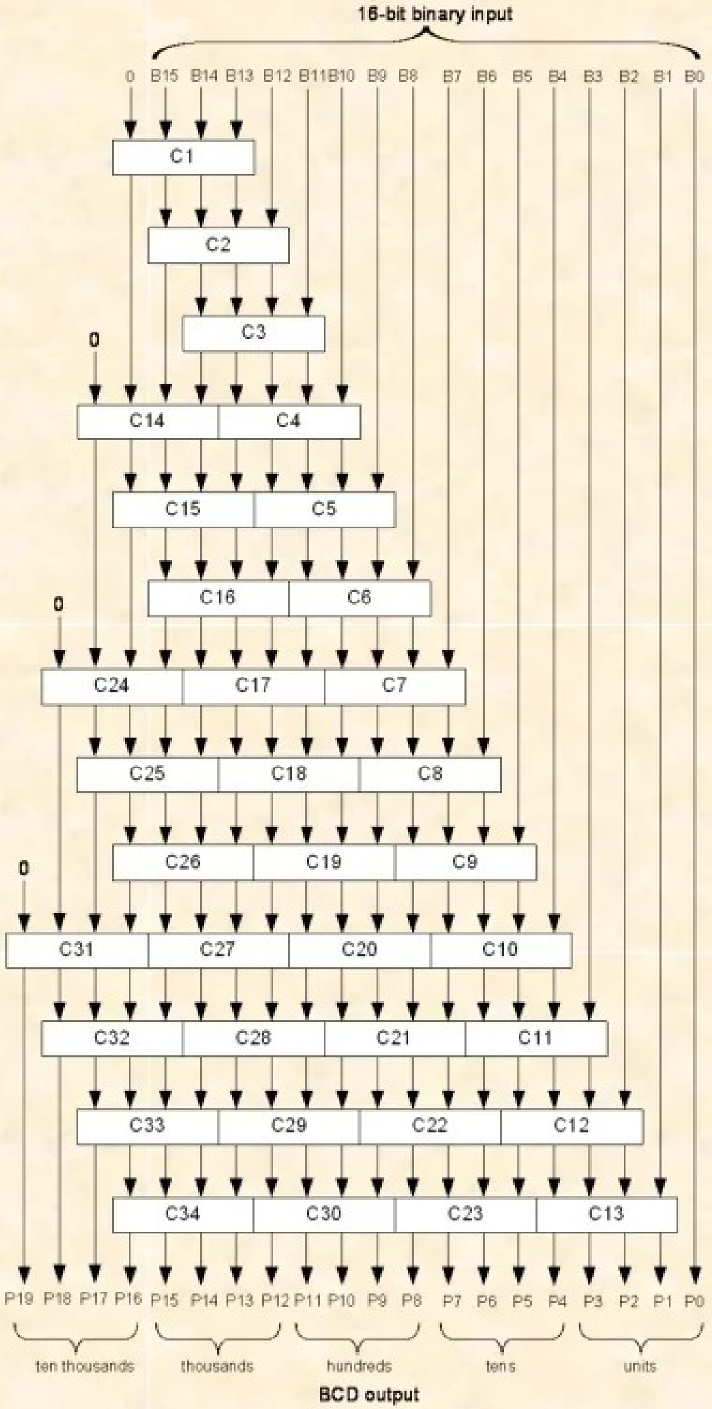


# Verilog *binbcd*

```
module binbcd(B,P);
       input [15:0] B;
       output [15:0] P;

       reg [15:0] P;
       reg [31:0] z;
       integer i;
```



```
  always @(B)
  begin
    for(i = 0; i <= 31; i = i+1)
       z[i] = 0;
    z[18:3] = B;

    for(i = 0; i <= 12; i = i+1)
    begin
       if(z[19:16] > 4)
              z[19:16] = z[19:16] + 3;
       if(z[23:20] > 4)
              z[23:20] = z[23:20] + 3;
       if(z[27:24] > 4)
              z[27:24] = z[27:24] + 3;
       if(z[31:28] > 4)
              z[31:28] = z[31:28] + 3;
       z[31:1] = z[30:0];
    end
    P = z[31:16];
  end
endmodule
```



# Steps to convert a 6-bit binary number to BCD

1. Clear all bits of *z* to zero
2. Shift *B* left 3 bits

    $z[8:3] = B[5:0];$

3. Do 3 times

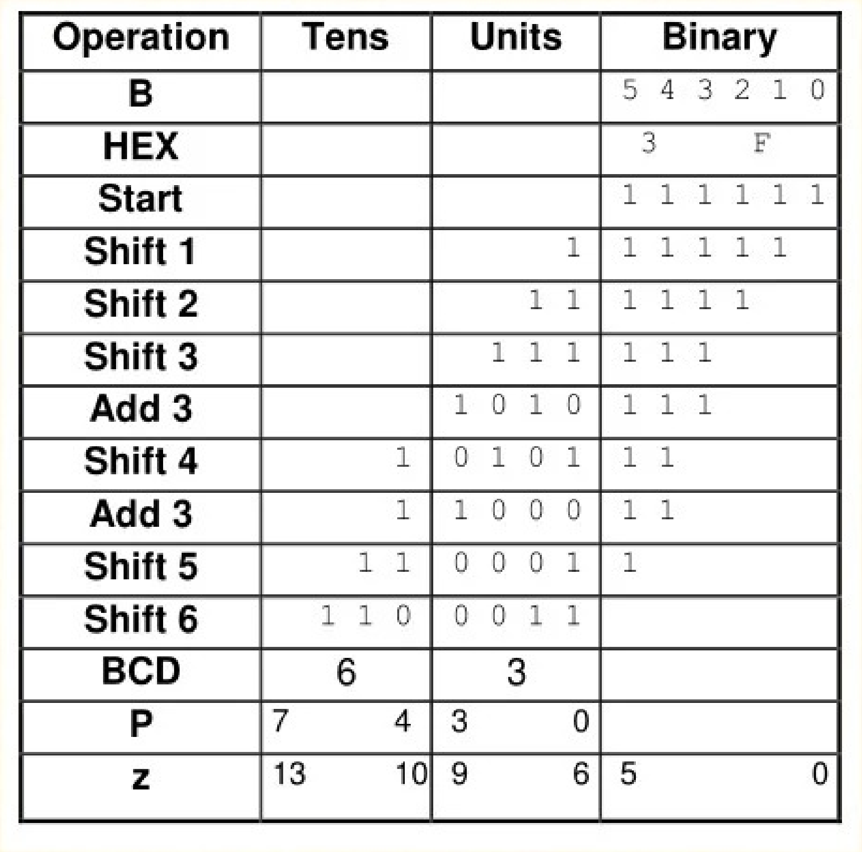
    if *Units* >4 then add 3 to *Units*

    (note: *Units* = $z[9:6]$)

    Shift *z* left 1 bit

4. *Tens* = $P[6:4] = z[12:10]$

    *Units* = $P[3:0] = z[9:6]$



| Operation | Tens | Units | Binary |
|---|---|---|---|
| B | | | 5 4 3 2 1 0 |
| HEX | | | 3 F |
| Start | | | 1 1 1 1 1 1 |
| Shift 1 | | 1 | 1 1 1 1 1 |
| Shift 2 | | 1 1 | 1 1 1 1 |
| Shift 3 | | 1 1 1 | 1 1 1 |
| Add 3 | | 1 0 1 0 | 1 1 1 |
| Shift 4 | 1 | 0 1 0 1 | 1 1 |
| Add 3 | 1 | 1 0 0 0 | 1 1 |
| Shift 5 | 1 1 | 0 0 0 1 | 1 |
| Shift 6 | 1 1 0 | 0 0 1 1 | |
| BCD | 6 | 3 | |
| P | 7 4 | 3 0 | |
| z | 13 10 | 9 6 | 5 0 |

Example of Verilog code for 16-bit conversion is shown on Slide 9

Example of VHDL code for 8-bit conversion is shown on Slide 6