Building our Digital World: Computer Systems & Architecture

# 1. 16-bit Counter HDL

The task is to produce an HDL file for a 16-bit counter with two inputs, 'inc' and 'reset'.  
When 'inc' is high, the counter will increment by one. When 'reset' is high, the counter will reset to zero.  
The HDL file below implements this counter using the Inc16 chip:  
  
CHIP Counter {  
 IN inc, reset;  
 OUT out[16];  
   
 PARTS:  
 // Initialize the Inc16 chip to create a 16-bit counter  
 Inc16(in=inc, load=reset, out=out);  
}

# 2. 4-bit Counter Driving a 7-Segment Display

The task is to implement a counter that counts from 0 to 15 (4-bit) and drives a 7-segment display.  
The counter has two inputs, 'inc' and 'reset', and the output is displayed on a 7-segment display.  
  
The HDL file below implements this counter using the Inc16 chip and a decoder to drive the display:  
  
CHIP DisplayCounter<group number> {  
 IN inc, reset;  
 OUT a, b, c, d, e, f, g;  
   
 PARTS:  
 // Initialize the Inc16 chip to create a 4-bit counter  
 Inc16(in=inc, load=reset, out=counter);  
   
 // Use a decoder to drive the 7-segment display based on the counter value  
 DecTo7Seg(in=counter[3..0], out={a, b, c, d, e, f, g});  
}

# 3. 8-bit Counter Driving Two 7-Segment Displays

The task is to adapt the previous counter to count from 0 to 255 (8-bit) and display the output using two 7-segment displays.  
The HDL file below implements this using two DecTo7Seg decoders to display the high and low 4 bits on two separate displays:  
  
CHIP MultiDisplayCounter<group number> {  
 IN inc, reset;  
 OUT DigitA[7], DigitB[7];  
   
 PARTS:  
 // Initialize the Inc16 chip to create an 8-bit counter  
 Inc16(in=inc, load=reset, out=counter);  
   
 // Use two decoders to drive two 7-segment displays  
 DecTo7Seg(in=counter[7..4], out=DigitA);  
 DecTo7Seg(in=counter[3..0], out=DigitB);  
}