Simplified floating-point adder Floating point is another format to represent a number. With the same number of bits, the range in floating-point format is much larger than that in signed integer format. Although VHDL has a built-in floating-point data type, it is too complex to be synthesized automatically. Detailed discussion of floating-point representation is beyond the scope of this book. We use a simplified 13-bit format in this example and ignore the round-off error. The representation consists of a sign bit, s, which indicates the sign of the number (1 for negative); a 4-bit exponent field, e, which represents the exponent; and an 8-bit significand field, f, which represents the significance of the fraction. In this format, the value of a floating-point number is (- 1)" * . f * 2". The . f * 2e is the magnitude of the number and (-1)' is just a formal way to state that "s equal to 1 implies a negative number." Since the sign bit is separated from the rest of the number, floating-point representation can be considered as a variation of the sign-magnitude format. We also make the following assumptions: Both exponent and significand fields are in unsigned format. The representation has to be either normalized or zero. Normalized representa- tion means that the MSB of the significand field must be 1. If the magnitude of the computation result is smaller than the smallest normalized nonzero magnitude, 0.10000000 * 20°00, it must be converted to zero. Under these assumptions, the largest and smallest nonzero magnitudes are 0.11 11 11 11 * 2"" and 0.10000000 * 2°000, and the range is about 2" (i.e., ~;$,AA~~~:$:::). Our floating-point adder design follows the process of add~ng numbers manually in scientific notation.