Chapter 5. Number Representation and Arithmetic Circuits

5.1 Number representation
- unsigned integers: representation and binary\(\leftrightarrow\)decimal conversion
- octal and hexadecimal representation and conversion

5.2 Addition of unsigned numbers
- single-bit addition: sum and carry bits
- half-adder truth table and logic circuit
- single-bit addition with carry-in (quickly derive or memorize)
  \[ s_i = x_i \oplus y_i \oplus c_i \quad c_{i+1} = x_i y_i + x_i c_i + y_i c_i \]
- full adder truth table and logic circuit
  \[ c = c_i + y_i \oplus x_i \quad s = x_i \oplus y_i \oplus c_i \]
- multi-bit addition: ripple-carry adder

5.3 Signed numbers
- sign-magnitude representation: sign bit
- 1’s complement representation: complement bits
- 2’s complement representation: 1’s complement + 1
  \[ \text{rule for fast computation: copy all 0s and first 1 from right; invert rest} \]
- subtraction using 2’s complement arithmetic
- adder symbol \(\sqrt{\oplus}\) with inputs and outputs (including carry-in and overflow)
- adder and subtractor construction from adder with XOR gates
- arithmetic overflow
- performance problems of ripple-carry

5.4 Fast adders
- carry propagate and generate (quickly derive or memorize)
  \[ c_{i+1} = g_i + p_i c_i \quad g_i = x_i y_i \quad p_i = x_i + y_i \]
- carry-lookahead adder logic circuit

5.7 Other representations
- fixed point numbers, radix point, and alignment
- floating point numbers: \(\pm M \times R^{ze} \)
  \[ \text{single precision format (memorize)} \]
- BCD representation and conversion
- ASCII representation
  \[ \text{letters, numbers, punctuation, and control symbols} \]
  \[ \text{(you do not need to memorize ASCII values for these)} \]
  \[ \text{parity: generation and checking of even and odd parity} \]
Chapter 6. Combinational-Circuit Building Blocks

6.1 Multiplexors
- \( n \)-to-1 functionality and construction from gates
- design of multiplexors from transmission gates and tri-state buffers
- construction of larger multiplexors from smaller ones
- synthesis of arbitrary logic functions
- majority function

6.2 Decoders and demultiplexors
- \( n \)-to-\( 2^n \) binary decoders functionality and construction from gates
- construction of larger decoders from smaller ones using enable input
- construction of larger decoders using decoder tree of smaller ones
- construction of multiplexor from decoder using additional gates or tri-state buffers
- 1-to-\( n \) demultiplexor construction and functionality from gates

6.3 Encoders
- \( 2^n \)-to-\( n \) binary encoder functionality and construction from gates
- priority encoder functionality and construction from gates

6.4 Code converters
- functionality
- BCD code converters

6.5 Arithmetic comparison
- equality comparison using \( \overline{XOR} \) gates
- magnitude comparison using gates (\( = \), \( > \), \( < \))