2’s Complement Representation
Binary Addition

A pair of binary numbers can be added bit by bit from right to left according to the same method used in decimal addition.

4-bit addition

\[
\begin{array}{c}
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
+ & 0 & 1 & 0 & 1 \\
\hline
0 & 1 & 1 & 1 & 0
\end{array}
\quad
\begin{array}{c}
1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
+ & 0 & 1 & 1 & 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 0
\end{array}
\]

no overflow
overflow

Computer hardware for binary addition of two n-bit numbers can be built from logic gates designed to calculate the sum of three bits (pair of bits plus carry bit).
Half Adder

- Adds two bits \(x\) and \(y\)
  - \(\text{sum} = x \oplus y\)
  - \(\text{carry} = x \cdot y\)

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
<th>carry</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

x
---
y

\[\text{xor} \quad \text{sum}\]

\[\text{and} \quad \text{carry}\]

\[\text{half adder} \quad \text{sum}\]

\[\text{carry}\]
Full Adder

- Adds three bits including a carry bit
- $\text{sum} = (x \oplus y) \oplus \text{cin}$
- $\text{carry} = x \cdot y + x \cdot \text{cin} + y \cdot \text{cin}$
Adder

- Adds two n-bit binary numbers
- n-bit ripple adder

\[ \begin{align*}
&\text{full adder} & \text{full adder} & \text{full adder} \\
&x_n & y_n & x_{n-1} & y_{n-2} & x_0 \\
&\text{cout} & \text{cin} & \text{sum} & \text{cout} & \text{cin} & \text{sum} \\
&c_n & s_{n-1} & c_{n-1} & c_{n-2} & c_1 & s_0
\end{align*} \]
Signed Binary Numbers

- A binary system with \( n \) bits can generate \( 2^n \) different bit patterns.
- To encode signed numbers in binary code, split this space into two equal subsets: positive number and negative numbers.
  - Keep hardware implementation as simple as possible.
    - 2’s complement representation
      - Used today by almost all computers.
2’s Complement Representation

- A.k.a. radix complement method
- The 2’s complement of a number $x$ in $n$-bit representation

$$\bar{x} = \begin{cases} 2^n - x & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

- $2^n$ signed numbers
  - Max = $2^{n-1} - 1$, Min = $-2^{n-1}$
- MSB is the sign bit
- Leave all the trailing 0’s and the fist least significant 1 intact, then flip all the remaining bits

  - Flip all the bits of $x$ and add 1 to the result
    - $0110 \rightarrow 1010$
Addition of Two Signed Numbers in 2’s Complement

- Exactly the same as the addition of unsigned binary numbers
  - $4 + (-5) = -1$
  - 4-bit addition
    - $(5) \ 0101 \rightarrow 1011 \ (-5)$

- Subtraction can be handled by converting it to addition
  - $x - y = x + (-y)$

- 2’s complement representation facilitates the addition of any two signed numbers without requiring special hardware
  - Need only a simple bit-wise adder
Decoder

- A decoder converts binary information from the n coded inputs to a maximum of $2^n$ unique outputs
- n-to-m-line decoders
  - $m \leq 2^n$
- When enable = 1, the decoder is enabled

<table>
<thead>
<tr>
<th>enable</th>
<th>a1</th>
<th>a0</th>
<th>d0</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

2-to-4 decoder

- a0
- d0
- a1
- d1
- enable
- d2
- d3
Decoder Expansion

- It is possible to combine two or more decoders to construct a bigger decoder
- A 3-to-8 decoder constructed with two 2-to-4 decoders
Sequential Logic
Combinational Logic vs. Sequential Logic

- **Combinational logic**
  - Input values are combined by the logical (Boolean) operations

- **Sequential logic**
  - Input values are stored over time as well as being combined
  - Maintains state
  - Equipped with memory elements

![Diagram of combinational vs. sequential logic](chart.png)
Clock

- A clock delivers a continuous train of alternating signals
  - An oscillator
  - Each clock cycle models one discrete time unit
- The clock signal is simultaneously broadcast to every sequential chip throughout the digital system
- Synchronisation
A binary storage device that is capable of storing one bit of information

**Data flip-flop**

- in: a data input, c: a clock input, out: a data output
- \( \text{out}(t) = \text{in}(t - 1) \) where \( t \) is the current clock cycle
Register

- A storage device that can store or remember a value over time
  - \[ \text{out}(t) = \text{out}(t - 1) \]
  - To store a value, put the value in the input and set load to 1
    - In the next clock cycle, the register commits to the new data value

- An n-bit register
  - An array of n single-bit registers
  - The multi-bit contents of such registers are typically referred to as words
Memory

- Random access memory (RAM) unit
  - Read/write operations on a RAM are able to access randomly chosen words, with no restrictions on the order in which they are accessed
- An array of n w-bit registers + some access logic
  - Three inputs: a data input, an address input, and a load bit
    - The address specifies which register should be accessed
    - Read when load = 0
    - Write when load = 1
Basic Design Parameters of RAM

- **Size**
  - The number of registers (words)

- **Width**
  - The width of each register (word)

- **Size \times Width RAM**
  - 64K \times 8 RAM, 256K \times 8 RAM, 16K \times 16 RAM, etc.
Counter

- A register that goes through a predetermined sequence of states upon the application of clock pulses
  - Typically $\text{out}(t) = \text{out}(t-1) + 1$
- reset
  - When reset = 1, it is reset to 0
- inc
  - When inc = 1, it increments its state in every clock cycle
Counter (contd.)

The diagram illustrates an n-bit counter with inputs and outputs labeled as follows:

- **Load**: 24
- **Reset**: 0
- **Inc**: 1
- **In**: 36
- **Clock**: 36

The counter's output sequence is as follows:

- **Out**: 24, 24, 0, 0, 1, 2, 36, 37

The diagram also shows the progression of the counter over a series of clock cycles, with each cycle incrementing the counter by 1 from 36 to 37.