1. Transitions of all signals are instantaneous.
2. Combinational gates: \( t_{pd} = t_{cont} = 0 \).
3. Flip-flops satisfy:
   \[
   t_{su} = t_{i+1} - t_i,
   \]
   \[
   t_{hold} = t_{cont} = t_{pd} = 0.
   \]
4. Simplified model for specifying and simulating the functionality of circuits with flip-flops.
1. The clock period, in the delay model, equals 1.
2. $t_{i+1} - t_i = 1$, for every $i$.
3. The duration of the $i$th clock cycle is the interval $[t_i, t_{i+1}) = [i, i + 1)$.
4. All transitions are instantaneous, so we may assume that each signal is stable during each clock cycle.
5. Let $X_i$ denote the digital value of the signal $X$ during the $i$’th clock cycle.
The functionality of a flip-flop is specified as follows:

\[ Q(t) = D(t - 1). \]

Since each signal is stable during each clock cycle, we could also write \( Q_i = D_{i-1} \).

meaning:

- The critical segment \( C_i \) equals \([t_{i-1}, t_i)\).
- The value of \( D(t) \) is stable during the critical segment \([t_{i-1}, t_i)\).
- This value, denoted by \( D_{i-1} \), is sampled by the flip-flop during the clock cycle \((i - 1)\).
- In the next clock cycle \([t_i, t_{i+1})\), the flip-flop’s output \( Q(t) \) equals the value of the input sampled during the previous cycle.
Example: Sequential XOR

- **Diagram:**
  - DFF (D Flip-Flop) with inputs A and Y.
  - XOR gate with inputs A and Z.
  - Clock (CLK) input.
  - Outputs Y and Z.

- **Table:**

<table>
<thead>
<tr>
<th>i</th>
<th>A_i</th>
<th>Y_i</th>
<th>Z_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
A term register is used to define a memory device that stores a bit or more. There are two main types of register depending on how their contents are loaded.

1. Parallel Load Register
2. Shift Register (also called a serial load register)
An *n*-bit parallel load register is specified as follows.

**Inputs:**
- $D[n - 1 : 0](t)$,
- $CE(t)$, and
- a clock $CLK$.

**Output:** $Q[n - 1 : 0](t)$.

**Functionality:**

\[
Q[n - 1 : 0](t + 1) = \begin{cases} 
D[n - 1 : 0](t) & \text{if } CE(t) = 1 \\
Q[n - 1 : 0](t) & \text{if } CE(t) = 0.
\end{cases}
\]

An *n*-bit parallel load register is simply built from *n* clock enabled flip-flops.
Figure: A 4-bit parallel load register
Parallel Load Register - simulation

### Diagram

![Parallel Load Register Diagram](image)

### Table

<table>
<thead>
<tr>
<th>$i$</th>
<th>$D[3:0]$</th>
<th>CE</th>
<th>$Q[3:0]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1010</td>
<td>1</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0101</td>
<td>1</td>
<td>1010</td>
</tr>
<tr>
<td>2</td>
<td>1100</td>
<td>0</td>
<td>0101</td>
</tr>
<tr>
<td>3</td>
<td>1100</td>
<td>1</td>
<td>0101</td>
</tr>
<tr>
<td>4</td>
<td>0011</td>
<td>1</td>
<td>1100</td>
</tr>
</tbody>
</table>
A *shift register* of $n$ bits is defined as follows.

**Inputs:** $D[0](t)$ and a clock $CLK$.

**Output:** $Q[n - 1](t)$.

**Functionality:** $Q[n - 1](t + n) = D[0](t)$.
Figure: A 4-bit shift register. Functionality: $Q[3](t + 4) = D[0](t)$
Shift Registers - simulation

<table>
<thead>
<tr>
<th>$i$</th>
<th>$D[0]$</th>
<th>$Q[3:0]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0011</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0111</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1110</td>
</tr>
</tbody>
</table>
Random Access Memory (RAM)

1. Array of memory cells.
2. Each memory cell stores a single bit.
3. In each cycle, a single memory cell is accessed.
4. Two operations are supported: read and write.
   - read operation: the contents of the accessed memory is output.
   - write operation: a new value is stored in the accessed memory.
5. The number of memory cells is denoted by $2^n$.
6. Each cell has a distinct address between 0 and $2^n - 1$.
7. The cell to be accessed is specified by an $n$-bit string called Address.
8. The array of memory cells is denoted by $M[2^n - 1 : 0]$. Let $M[i](t)$ denote the value stored in the $i$th entry of the array $M$ during clock cycle $t$. 
Definition

A RAM($2^n$) is specified as follows.

Inputs: $Address[n - 1 : 0](t) \in \{0, 1\}^n, D_{in}(t) \in \{0, 1\}, \ R/\overline{W}(t) \in \{0, 1\}$ and a clock $CLK$.

Output: $D_{out}(t) \in \{0, 1\}$.

Functionality:

1. data: array $M[2^n - 1 : 0]$ of bits.
2. initialize: $\forall i : M[i] \leftarrow 0$.
3. For $t = 0$ to $\infty$ do
   1. $D_{out}(t) = M[\langle Address \rangle](t)$.
   2. For all $i \neq \langle Address \rangle$: $M[i](t + 1) \leftarrow M[i](t)$.
   3. $M[\langle Address \rangle](t + 1) \leftarrow \begin{cases} D_{in}(t) & \text{if } R/\overline{W}(t) = 0 \\ M[\langle Address \rangle](t) & \text{else.} \end{cases}$
Figure: A schematic of a \( \text{RAM}(2^n) \).
RAM -design

Address\([n-1:0]\)

DECODER\(n\)

\(B[2^n-1:0]\)

\(D_{in}\)

\(B[2^n-1]\)

\(M_{2^n-1}\)

\(R/\overline{W}\)

\(D[2^n-1]\)

\(D_{in}\)

\(B[1]\)

\(M_1\)

\(R/\overline{W}\)

\(D[1]\)

\(D_{in}\)

\(B[0]\)

\(M_0\)

\(R/\overline{W}\)

\(D[0]\)

\(D[2^n-1:0]\)

\(2^n\)

\(D_{out}\)

\(D[2^n-1:0]\)

\(Address[\(n-1:0\)]\)

\(n\)

\((2^n:1) - MUX\)

\(n\)
## Memory Cell - specification

### Definition

A single bit *memory cell* is defined as follows.

**Inputs:** $D_{\text{in}}(t)$, $R/W(t)$, $sel(t)$, and a clock $\text{CLK}$.

**Output:** $D_{\text{out}}(t)$.

### Functionality:

Let $S(t) \in \{0, 1\}$ denote the state of memory cell in cycle $t$. Assume that the state is initialized to be $S(0) = 0$. The functionality is defined according to the following cases.

1. $S(t) \leftarrow \begin{cases} D_{\text{in}}(t) & \text{if } sel(t) = 1 \text{ and } R/W(t) = 0 \\ S(t-1) & \text{otherwise.} \end{cases}$

2. $D_{\text{out}}(t) \leftarrow S(t-1)$. 

Figure: An implementation of a memory cell.
The module called Read-Only Memory (ROM) is similar to a RAM, except that write operations are not supported. This means that the contents stored in each memory cell are preset and fixed. ROMs are used to store information that should not be changed. For example, the ROM stores the program that is executed when the computer is turned on.
A ROM($2^n$) that implements a Boolean function $M : [0..2^n − 1] \to \{0, 1\}$ is defined as follows.

**Inputs:** $Address[n − 1 : 0](t)$.

**Output:** $D_{out}(t)$.

**Functionality:**

$$D_{out} = M[\langle Address \rangle].$$