0.1 Design Examples

0.1.1 Three-Way Light Control

- Say there is a large room with 3 lights, with corresponding switches. Let x_1, x_2, x_3 denote the state of each switch.
- The light is OFF when all switches are open. The light is ON if exactly one switch is closed, and OFF if two (or no) switches are closed.
- Let state of the light be represented by the function $f(x_1, x_2, x_3)$.

We shall try to represent$f(x_1, x_2, x_3)$ in sum-of-products form
Sum-of-Products Form
The function f is expressed as:

$$f = m_1 + m_2 + m_4 + m_7 = \overline{x_1}x_2\overline{x_3} + x_1\overline{x_2x_3} + \overline{x_1x_2}x_3 + x_1x_2x_3$$

This expression cannot be simplified further.

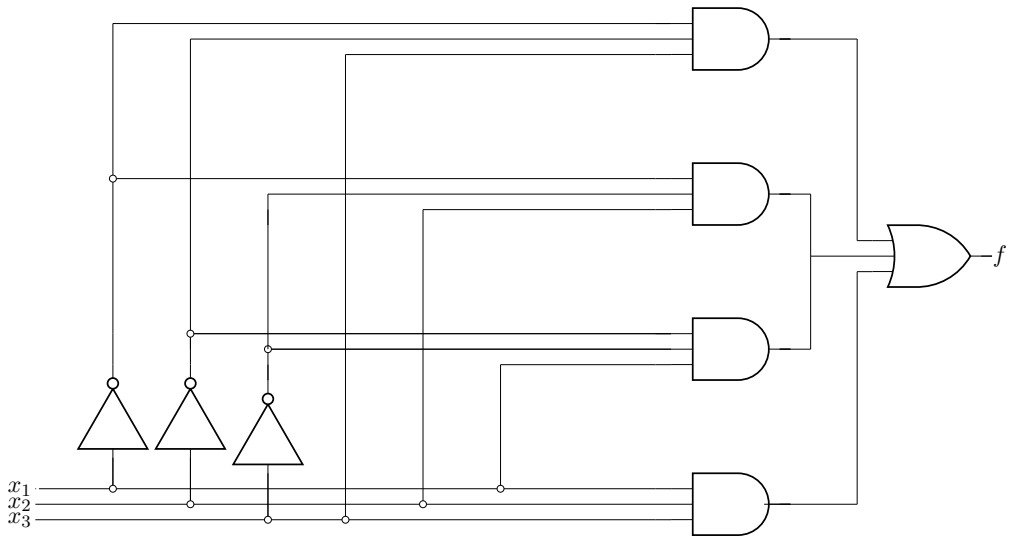


Figure 1: Sum-of-Products

Product-of-Sums Form
The function f can also be expressed as:

$$f = M_0 \cdot M_3 \cdot M_5 \cdot M_6 = (x_1 + x_2 + x_3)(x_1 + \overline{x_2} + \overline{x_3})(\overline{x_1} + x_2 + \overline{x_3})(\overline{x_1} + \overline{x_2} + x_3)$$

This form has the same cost as the sum-of-products realization.

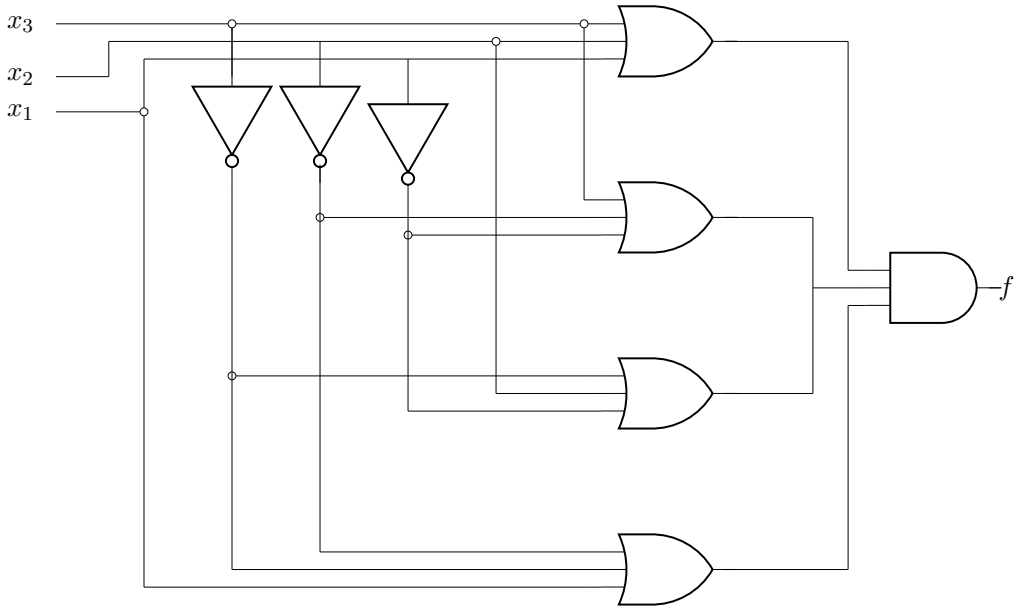


Figure 2: Product-of-Sums

Truth Table
Truth table for the 3-way light control is as follows,

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

0.1.2 Multiplexer Circuit

- In computer systems, it is often necessary to select data from exactly one of several sources.Consider two data sources, represented by input signals x_1 and x_2, which change over time.

- A circuit is needed to produce an output that matches either x_1 or x_2, depending on a selection control signal s.

The circuit has three inputs: x_1, x_2, and s. The output f is defined as:

$$f = \begin{cases} x_1 & \text{if } s = 0, \\ x_2 & \text{if } s = 1. \end{cases}$$

The Sum-of-Products form of the function is,

$$f(s, x_1, x_2) = sx_1x_2 + sx_1\overline{x_2} + \overline{s}x_1x_2 + \overline{sx_1}x_2.$$

Using the distributive property,

$$f = \overline{s}x_1(\overline{x_2} + x_2) + s(\overline{x_1} + x_1)x_2$$

Applying theorems of boolean algebra $x + \overline{(x)} = 1$,

$$f = sx_1 + \overline{s}x_2.$$

- A circuit implementing this function is called a **2-to-1 multiplexer**. It selects one of two data inputs (x_1 or x_2) based on the control signal s.

- Multiplexers are widely used in digital systems, and larger multiplexers (e.g., 4-to-1, 8-to-1) can be built by extending this concept. A 4-to-1 multiplixer would require two control signals, while an 8-to-1 multiplexer would require three.

The truth table for the 2-to-1 multiplexer is as follows:

s	x_1	x_2	$f(s, x_1, x_2)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

A 2-to-1 multiplexer can be built using the basic AND, OR, NOT gates

0.2 Summary of CAD Tools for Logic Circuit Design

Logic circuits in complex systems, such as modern computers, are designed using sophisticated **CAD tools** rather than manual methods. These tools are packaged into a **CAD system**, which includes tools for:

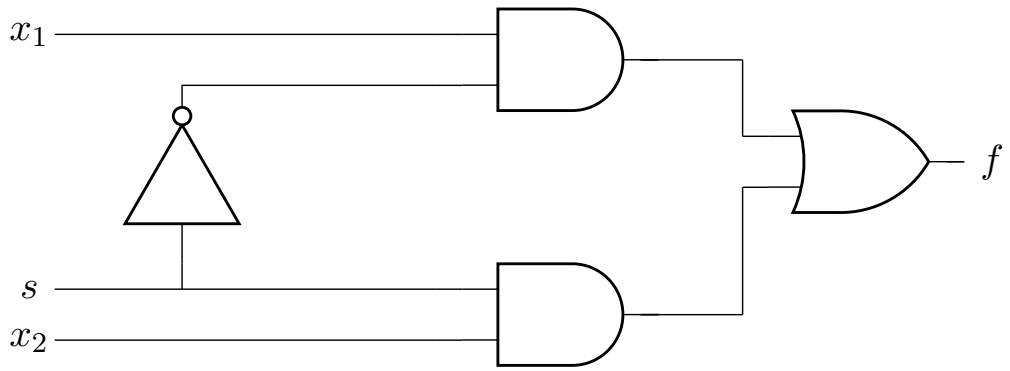


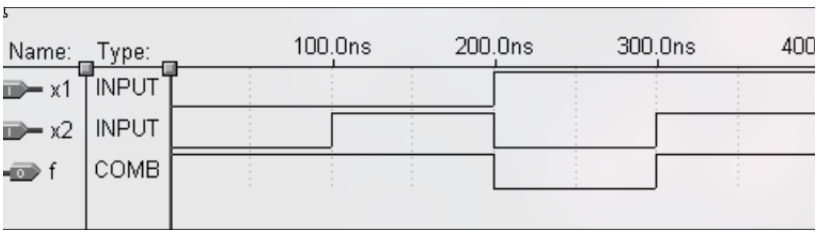
Figure 3: 2-to-1 Multiplexer

- Design entry
- Synthesis and optimization
- Simulation
- Physical design

0.2.1 Design Entry

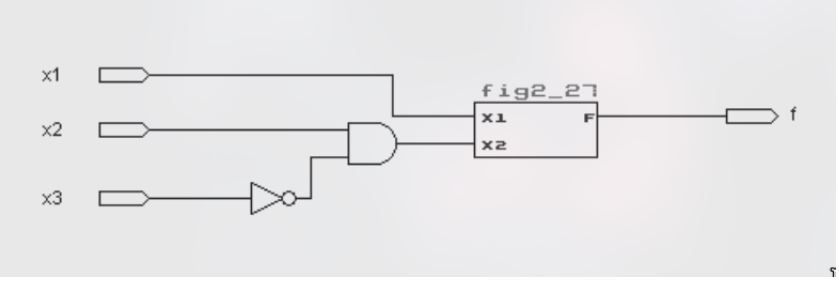
Design entry is the first step in the design process, where the circuit's functionality is described. Three common methods are:

1. **Truth Tables**: Suitable for small circuits or subcircuits. CAD tools can transform truth tables or even waveforms into logic gate networks. One example of a waveform being given as input is, It is not suitable



   for larger circuits.

2. **Schematic Capture**: Uses graphical symbols to represent logic gates and interconnections. Supports hierarchical design, where subcircuits are represented as symbols. One example of a circuit represented using Schematic Capture is, It is a much better way of representing larger circuits, but its drawbacks are differences in UI, functionality of different



software. This means repeated training when switching tools. Another drawback to note is that the GUI becomes awkward in case of larger circuits.

3. **Hardware Description Languages (HDLs)**: They are similar to computer programming languages except, HDL's are used to describe hardware rather than software. Two HDL's are IEEE standard - **Verilog** and **VHDL**. In comparission to Schematic capture, a HDL such as verilog provies *portability*. This means that a circuit specified in Verilog can be implemented in different types of chips and with CAD tools provided by different companies, without having to change the Verilog specification. Both small and large logic circuit designs can be efficiently represented in Verilog code. Verilog has been used to define circuits such as microprocessors with millions of transistors. Verilog design entry can be combined with other methods. For example, a schematic- capture tool can be used in which a subcircuit in the schematic is described using Verilog.

0.2.2 Synthesis Process

Synthesis is the process of generating a logic circuit from a high-level description, such as a truth table, schematic, or Verilog code. CAD tools automate this process, which includes:

- Translating Verilog code or schematic diagrams into logic expressions.
- Generating logic equations from truth tables.
- Logic Optimization

Logic Optimization

The initial logic expressions provided by synthesis tools are often not optimal (reflecting on the designer's input). **Logic optimization** is a critical step where the synthesis tools manipulate these expressions to produce a more

efficient circuit. Optimization goals depend on the design requirements and the target hardware technology.

0.2.3 Functional Simulation

After completing design entry and synthesis, it is essential to verify that the designed circuit functions as expected. This is done using a **functional simulator**, which evaluates the circuit's behavior based on:

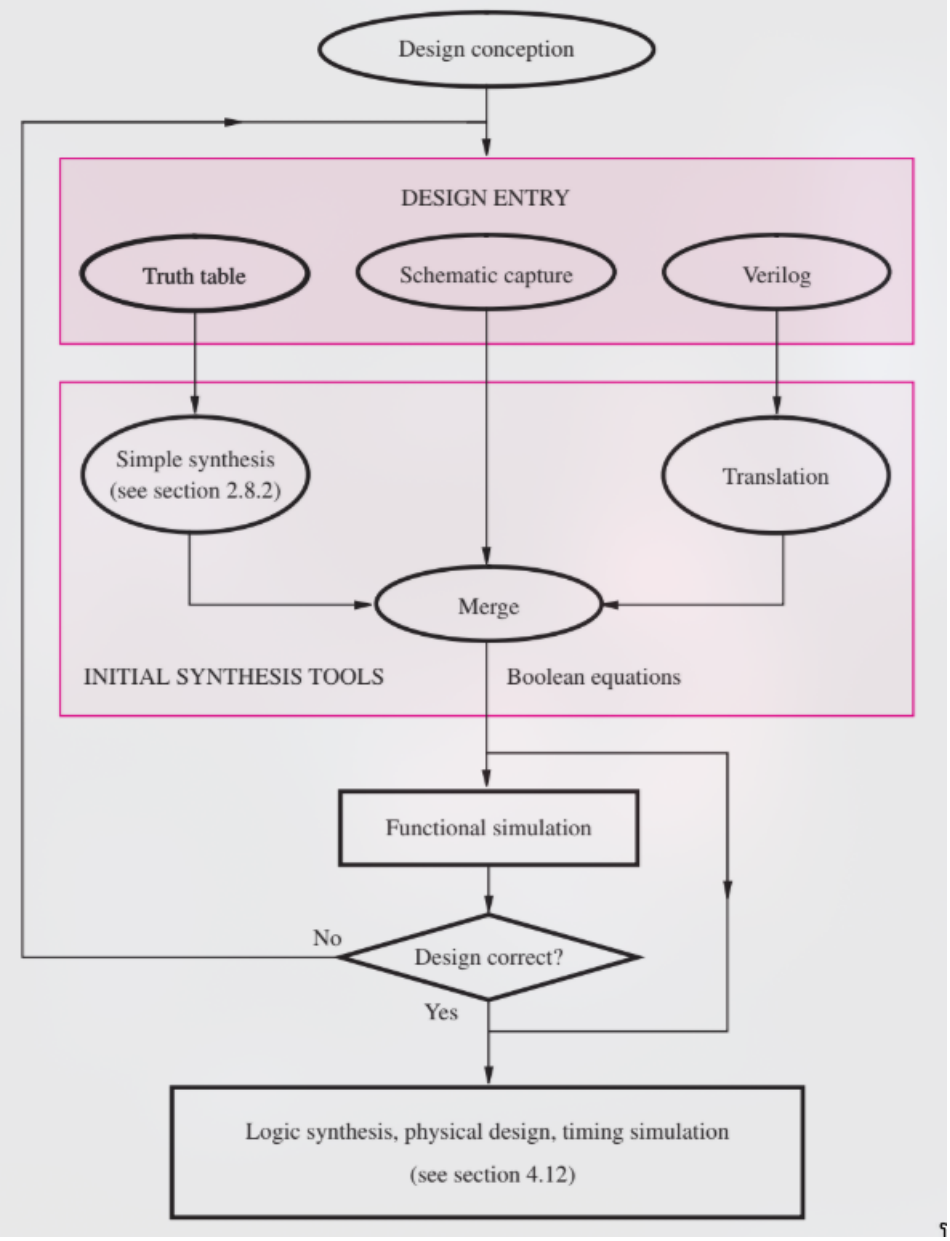
- Logic equations generated during synthesis (before optimization).
- Input valuations specified by the user.

The functional simulator:

- Applies input valuations to the logic equations.
- Computes the corresponding outputs.
- Presents the results as a truth table or timing diagram.

The user examines these results to ensure the circuit operates correctly. **Drawback:** The functional simulator assumes that the time needed for signals to propagate through the logic gates is negligible. In real logic gates this assumption is not realistic, regardless of the hardware technology chosen for implementation of the circuit.

0.2.4 Summary


Design conception
DESIGN ENTRY
Truth table
Schematic capture
Verilog
Simple synthesis
(see section 2.8.2)
Translation
Merge
INITIAL SYNTHESIS TOOLS
Boolean equations
Functional simulation
No
Design correct?
Yes
Logic synthesis, physical design, timing simulation
(see section 4.12)