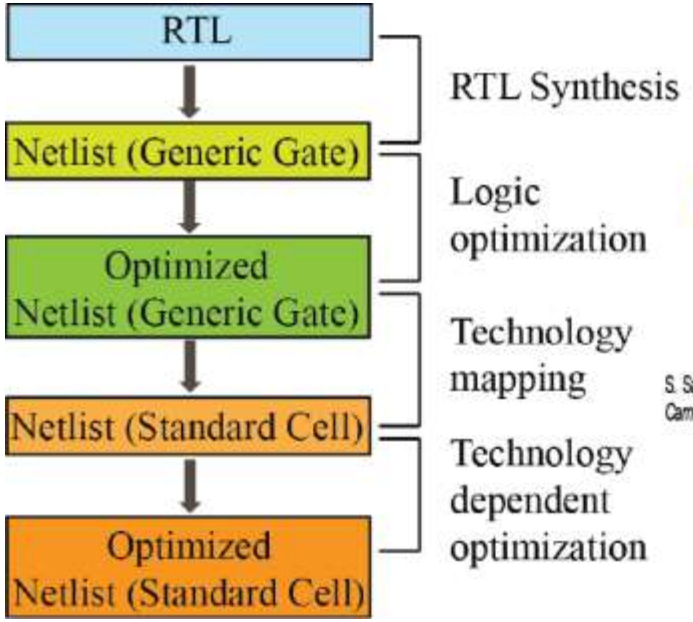
# VLSI DESIGN FLOW: RTL TO GDS

## Lecture 14
Logic Optimization: Part I

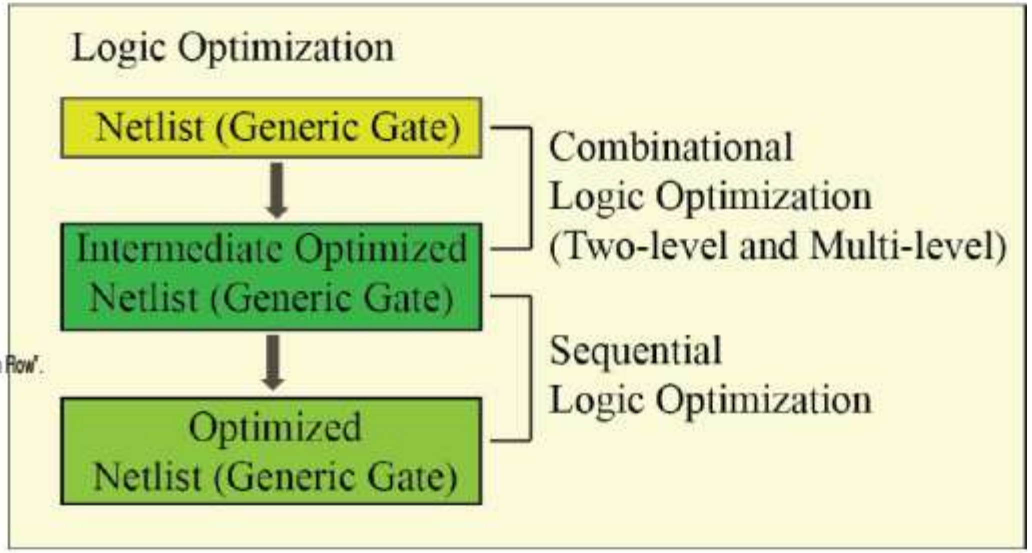


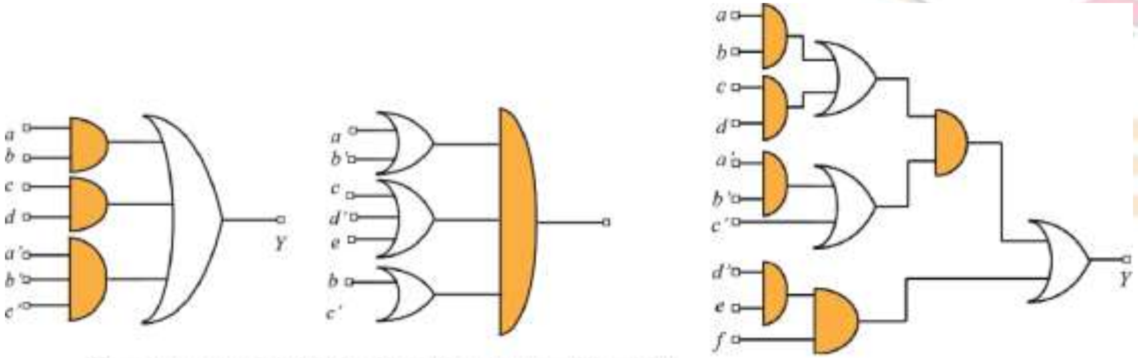
Sneh Saurabh
Electronics and Communications Engineering
IIIT Delhi

# Lecture Plan




S. Saurabh, "Introduction to VLSI Design Flow". Cambridge University Press, 2023.

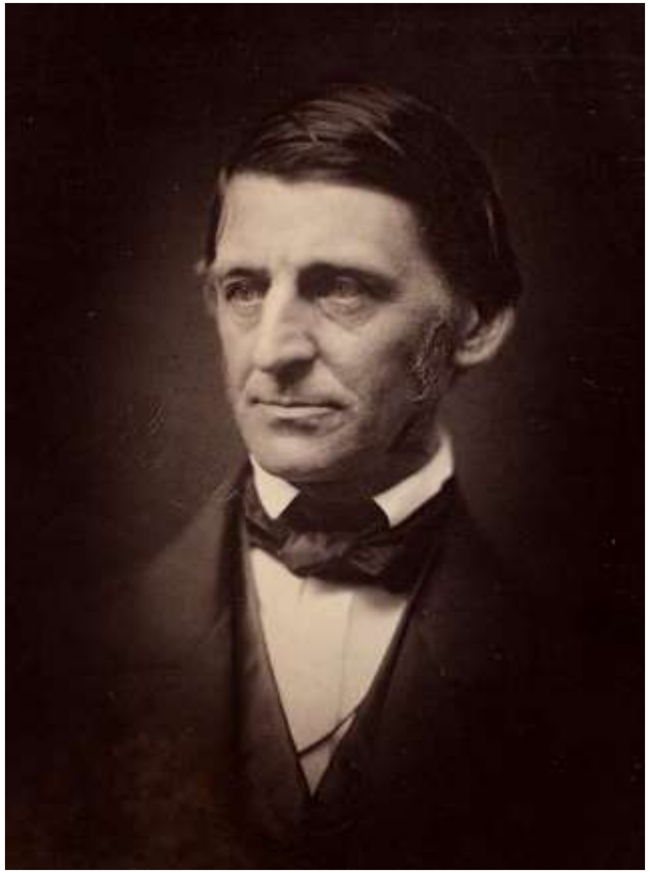


S. Saurabh, "Introduction to VLSI Design Flow". Cambridge University Press, 2023.

- Two-level Combinational Logic Optimization

# Simplicity and beauty ...



'We ascribe beauty to that which is simple; which has no superfluous parts; which exactly answers its end...'

—R. W. Emerson, *The Conduct of Life*, on "Beauty," 1871

Source: https://commons.wikimedia.org/wiki/File:Ralph_Waldo_Emerson_by_Josiah_Johnson_Hawes_1857.jpg Josiah Johnson Hawes, Public domain, via Wikimedia Commons

# Logic Optimization

Two-level

# Boolean Function: Definitions

**Boolean variable**: variable that can take one of the two values 0 or 1

**Boolean function**: function that takes Boolean variables as arguments and evaluates to 0 or 1.

Denoted as: $y = f(x_1, x_2, x_3, \ldots, x_N)$ where the variables $\{x_1, x_2, x_3, \ldots, x_N\}$ are Boolean variables.

**Literal:** a Boolean variable or its complement.

Example: $x_1, x_2, x_1', x_3'$ etc.

**Cube**: a product of literals.

Example: $x_1x_2x_3, x_1x_2', x_3'$ etc.

Consider a Boolean function of $N$ variables:

**Minterm:** the cube of $N$ literals in which each variable or its complement appears exactly once.

**Maxterm:** the sum of $N$ literals in which each variable or its complement appears exactly once.

**Example**: Consider a function of three variables $x_1, x_2, x_3$:

Minterms are: $x_1x_2x_3$, $x_1'x_2x_3$, $x_1'x_2'x_3$ etc.

Maxterms are $(x_1 + x_2 + x_3)$ , $(x_1 + x_2' + x_3)$, etc.

# Boolean Function Representations

**Truth Table:** The row of a truth table shows the value (0 or 1) assigned to each input variable $x_1, x_2, \dots, x_N$ and its corresponding output value.

- Will contain $2^N$ rows in the table.

**Minterm representation of a function:**

- Each row of a truth table corresponds to a minterm.
  - The minterm evaluates to 1 only for the assignment of variables corresponding to that row in the truth table.
  - For all other assignments of variables, the minterm evaluates to 0
- **Sum of minterms:** From the truth table, we take the sum of only those minterms for which the function evaluates to 1.

$$y = x_1x_2 + x_2x_3 + x_3x_1$$

| $x_1$ | $x_2$ | $x_3$ | $y$ | Minterm |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $x_1'x_2'x_3'$ |
| 0 | 0 | 1 | 0 | $x_1'x_2'x_3$ |
| 0 | 1 | 0 | 0 | $x_1'x_2x_3'$ |
| 0 | 1 | 1 | 1 | $x_1'x_2x_3$ |
| 1 | 0 | 0 | 0 | $x_1x_2'x_3'$ |
| 1 | 0 | 1 | 1 | $x_1x_2'x_3$ |
| 1 | 1 | 0 | 1 | $x_1x_2x_3'$ |
| 0 | 1 | 1 | 1 | $x_1x_2x_3$ |

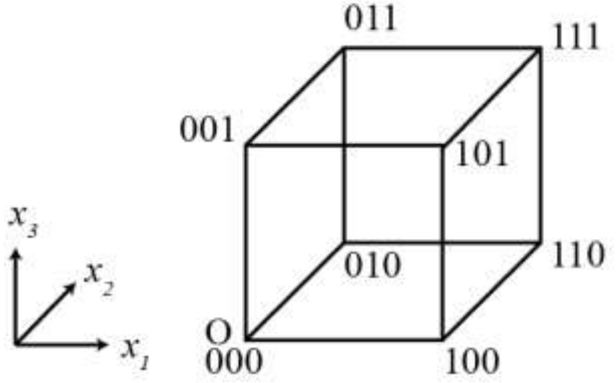
$$y = x_1'x_2x_3 + x_1x_2'x_3 + x_1x_2x_3' + x_1x_2x_3$$

# Boolean Space and Hypercube: Definition

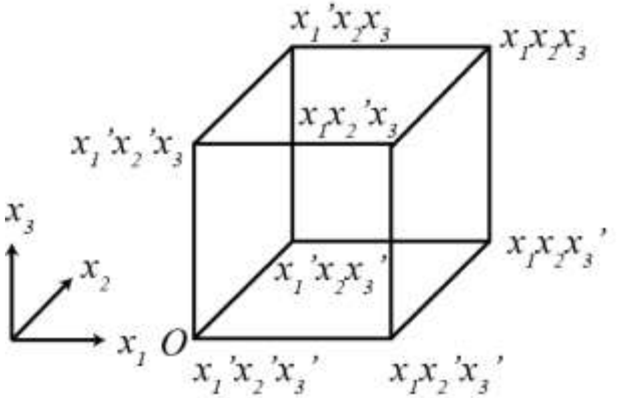
- Boolean functions of $N$ variables $[y = f(x_1,x_2,x_3,\ldots,x_N)]$ spans an $\boldsymbol{N}$ **–dimensional Boolean space**
- An $N$ –dimensional Boolean space can be represented and visualized using an $\boldsymbol{N}$ **–dimensional Boolean hypercube**

**Boolean Hypercube:**

- Associate each variable with one dimension of the hypercube.
- The corners of the hypercube represent binary-valued N-dimensional vectors
  - ➢ i-th entry in the vector corresponds to the value of variable $x_i$.
- A Boolean hypercube has $2^N$ corners, representing $2^N$ input combinations (or the associated minterms)




S. Saurabh, "Introduction to VLSI Design Flow". Cambridge University Press, 2023.



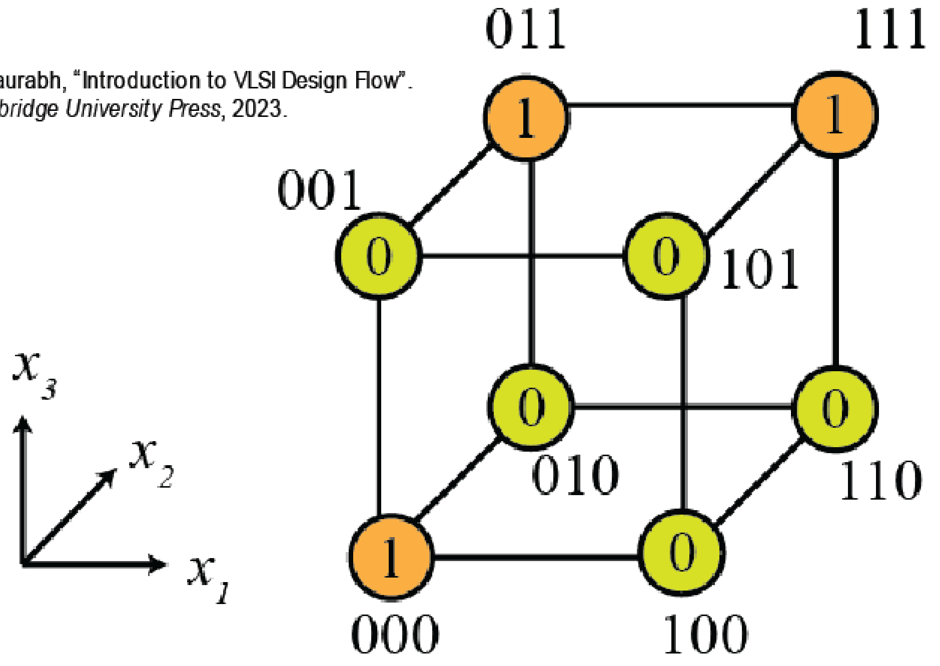

S. Saurabh, "Introduction to VLSI Design Flow". Cambridge University Press, 2023.

# Boolean Function Representation: Hypercube

- Mark the corners in the Boolean hypercube with the value of the function for the associated input combination.

$$y = x_1'x_2'x_3' + x_1'x_2x_3 + x_1x_2x_3$$

S. Saurabh, "Introduction to VLSI Design Flow". *Cambridge University Press*, 2023.



# Don’t Care (DC) Conditions

**Don’t Care (DC) conditions:**

- For some input combinations, the function $y = f(x_1, x_2, x_3, \ldots, x_N)$ may not be specified.
  - These input combinations are known as don’t care
- DC conditions are denoted by X

- DC conditions are related to the input combinations that can never occur
  - Example: If a function receives binary coded decimal (BCD) digits it cannot receive input combinations {1010, 1011, …, 1111}
- Can also be those input combinations for which the output is not observed
  - Example: A function producing an output for a block that is in a sleep state

# Incompletely-specified Boolean Function

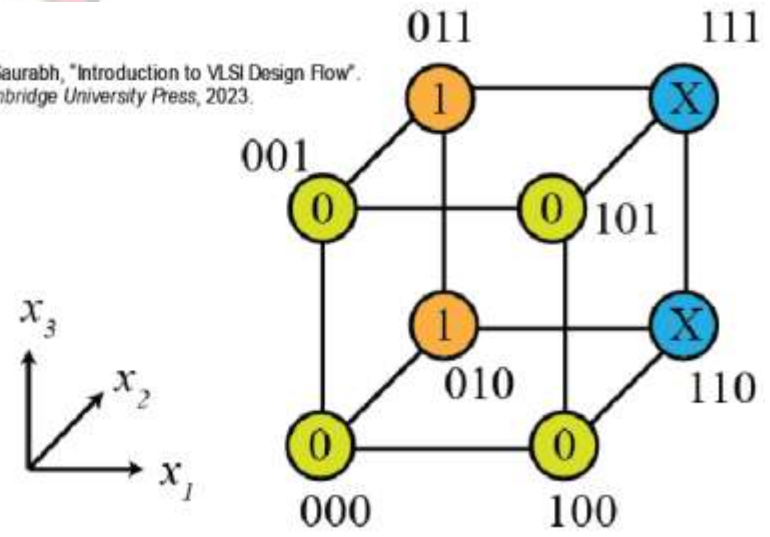
**Incompletely-specified Boolean function:**

- A Boolean function with DC conditions
- Can represent using three sets:
  - **ON-set**: input combinations for which the output is 1
  - **OFF-set**: input combinations for which the output is 0
  - **DC-set**: input combinations for which the output is X

**Example:**

- A function of three variables $x_1$,$x_2$, and $x_3$.
  ON-set={010, 011},
  OFF-set={000, 001, 100, 101}, and
  DC-set={110, 111}.

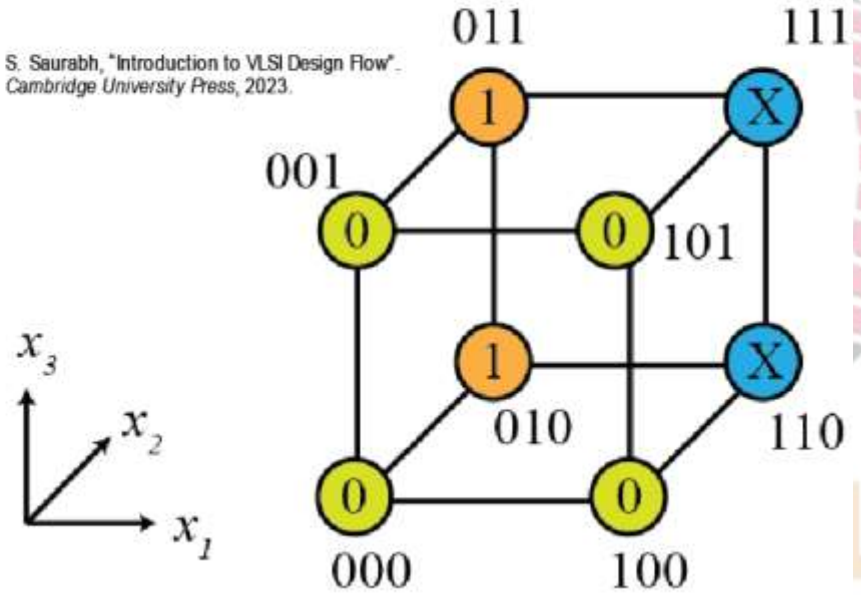
S. Saurabh, "Introduction to VLSI Design Flow". Cambridge University Press, 2023.



# Implicant of a Boolean Function

**Implicant of a Boolean function**:

- A cube whose corners are all in the ON-set or DC-set of that function




**Implicants:**

- $x_1'x_2x_3'$
- $x_1x_2$
- $x_2x_3'$
- $x_2$

**Not an implicant:**

- $x_1'x_3'$
- $x_1'$

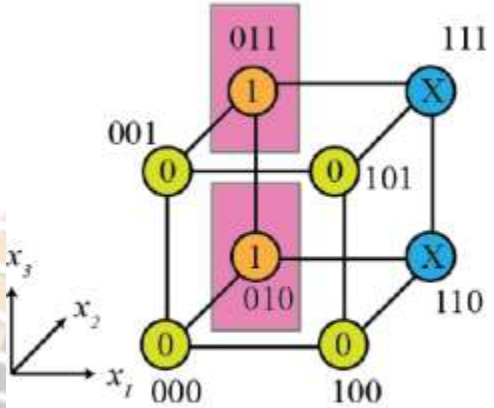
# Cover of a Boolean Function

**Cover of a Boolean function**:

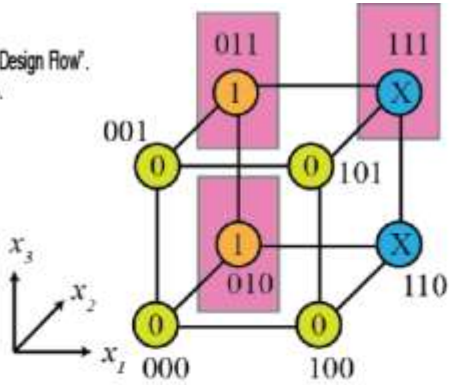
- Set of implicants that includes all its minterms

- The number of implicants in a cover is known as the **size of the cover**.

- The cover with the minimum size is known as the **minimum cover**

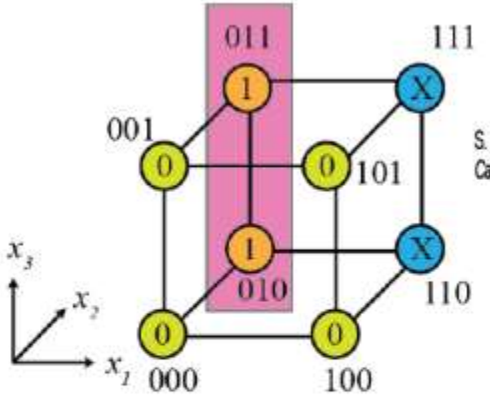



S. Saurabh, "Introduction to VLSI Design Flow". Cambridge University Press, 2023.

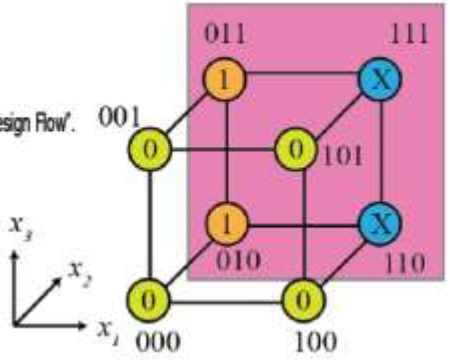



$$y = x_1'x_2x_3' + x_1'x_2x_3$$

$$y = x_1'x_2x_3' + x_1'x_2x_3 + x_1x_2x_3$$




S. Saurabh, "Introduction to VLSI Design Flow". Cambridge University Press, 2023.




$$y = x_1'x_2$$

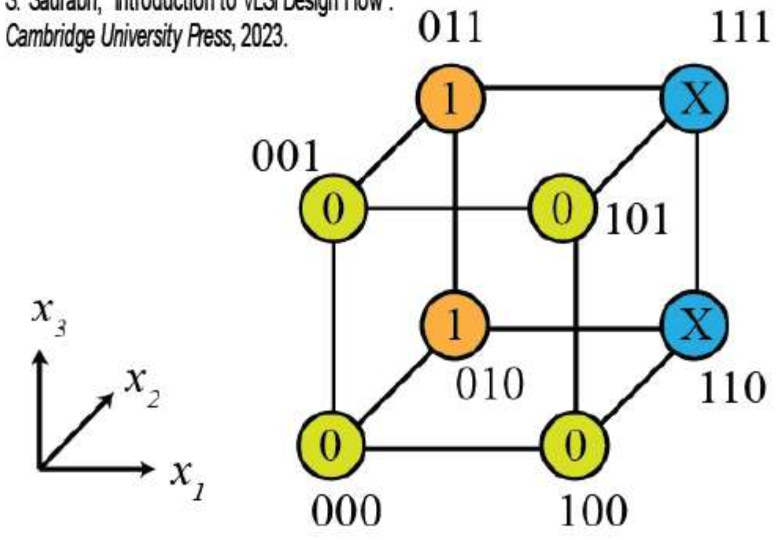
$$y = x_2$$

# Prime Implicant

**Prime Implicant of a function:**

- An implicant of a function that is not covered by any other implicant of that function
- Also called prime (in short)

S. Saurabh, "Introduction to VLSI Design Flow". *Cambridge University Press*, 2023.



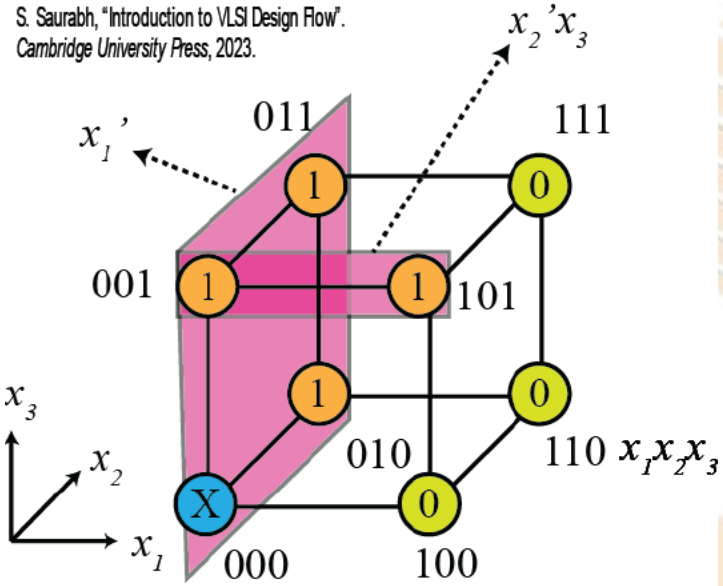

**Prime Implicants:**

- $x_2$: Yes
- $x_1'x_2x_3'$: No
- $x_1x_2$: No

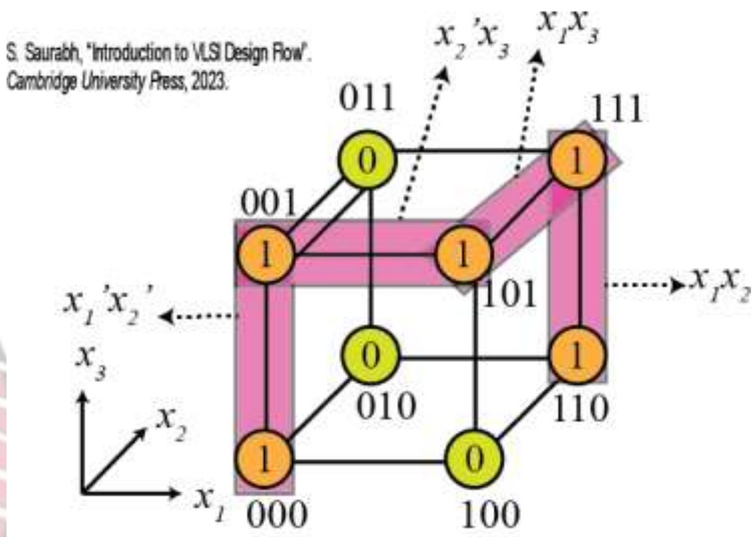
# Essential Prime Implicant and Prime Cover

**Essential prime implicant:**

- If there is at least one minterm that is covered by only that prime implicant.




- Both $x_1'$ and $x_2'x_3$ are essential primes




- **Prime implicants:** $x_1'x_2'$, $x_2'x_3$, $x_1x_3$, and $x_1x_2$
- **Essential primes:** $x_1'x_2'$ and $x_1x_2$
- **Non-essential primes:** $x_2'x_3$ and $x_1x_3$

# Exact Two-level Logic Minimization

- **Aim:** To find the minimum cover.
- **Reason:** correlates with the circuit's reduced hardware or reduced area.

- **For simple problem:** conventional techniques such as Boolean algebra-based manipulations and Karnaugh maps.
- **For problems with more than 20 variables:** conventional techniques becomes too complicated.

- In finding the minimum cover, we can reduce the search space by employing Quine's theorem

# Quine's Theorem

**Prime cover**: a cover that consists only of prime implicants

**Quine's Theorem:**

- There exists a minimum cover consisting only of prime implicants (prime cover)

**Proof:**

- Consider a minimum cover that is not a prime cover.
  - It implies that it contains some implicants that are not prime implicants.
  - Can replace each non-prime implicant with a prime implicant that contains it.
  - A new cover is obtained that consists of primes only and is of the same size.
  - Hence, there exists a minimum cover that is a prime cover.

**Application:**

- We can focus on only prime cover for finding minimum cover.

# Prime Implicant Table

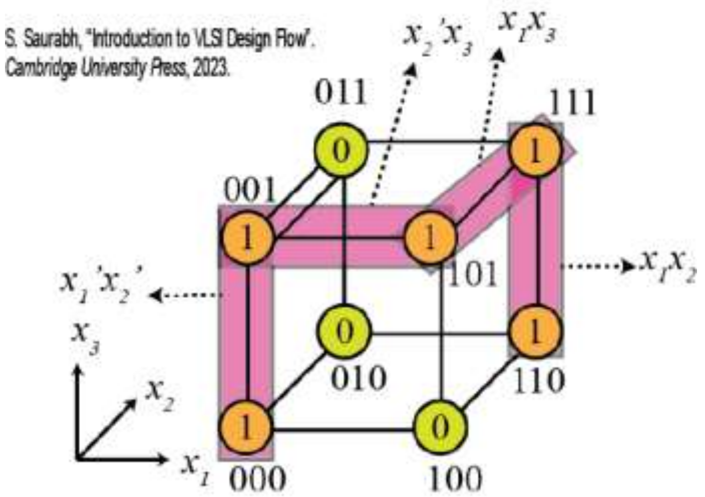
How to find minimum cover among prime covers?

- By finding the set of **prime implicants** and building a **prime implicant table**.

How to build prime implicant table?

- Arrange the prime implicants in separate columns and the minterms in individual rows of the prime implicant table.
- Fill entries in the table: If a given prime (column) covers a given minterm (row), then the corresponding entry is made 1, else it is made 0.

# Prime Implicant Table: Illustration




| | Prime Implicants | | | |
|---|---|---|---|---|
| Minterms | $x_1'x_2'$ | $x_2'x_3$ | $x_1x_3$ | $x_1x_2$ |
| $x_1'x_2'x_3'(000)$ | 1 | 0 | 0 | 0 |
| $x_1'x_2'x_3(001)$ | 1 | 1 | 0 | 0 |
| $x_1x_2'x_3(101)$ | 0 | 1 | 1 | 0 |
| $x_1x_2x_3'(110)$ | 0 | 0 | 0 | 1 |
| $x_1x_2x_3(111)$ | 0 | 0 | 1 | 1 |

**How to find minimum cover using prime implicant table?**

- Find the minimum set of columns that covers all the rows in the prime implicant table.
- Solve set covering problem (can grow exponentially with the number of variables)

**Simplification:**

- Identify essential primes: ($x_1'x_2'$ and $x_1x_2$)
- Work on remaining minterms [$x_1x_2'x_3(101)$]: cover either using $x_2'x_3$ or $x_1x_3$]
- Efficient reduction and covering algorithms

# Heuristic Minimizer: Basics

For large problems we prefer heuristic minimizer over exact minimization.

- Heuristic minimizers are faster for large problem sizes.
- We often do not need the exact minimum cover.
  - Any solution that can be found quickly and is near-optimal is acceptable.

**Minimal Cover:**

- A minimal cover satisfies certain local minimum cover property rather than the global minimum property.
- For example, a cover in which no implicant is contained in any other implicant of the cover.
  - Minimal with respect to single-implicant containment.

- The size of a minimal cover can be more than the size of the minimum cover.

# Heuristic Minimizer: Approach

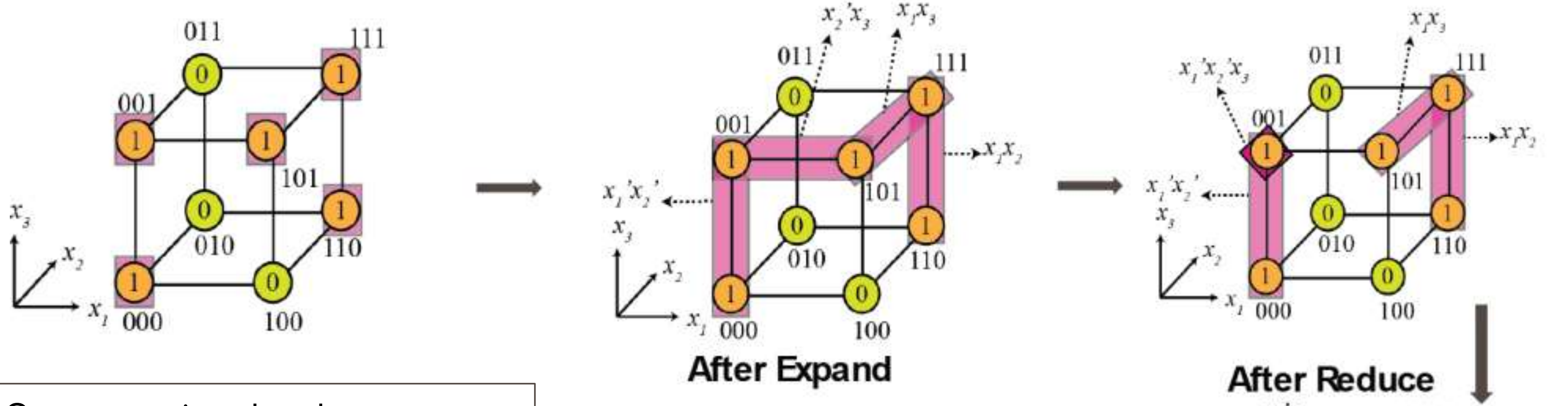
**Approach:**

- Starts with an initial cover and iteratively improves the solution by applying **some operators** on it.
- The iteration terminates when the algorithm can no longer improve the solution.

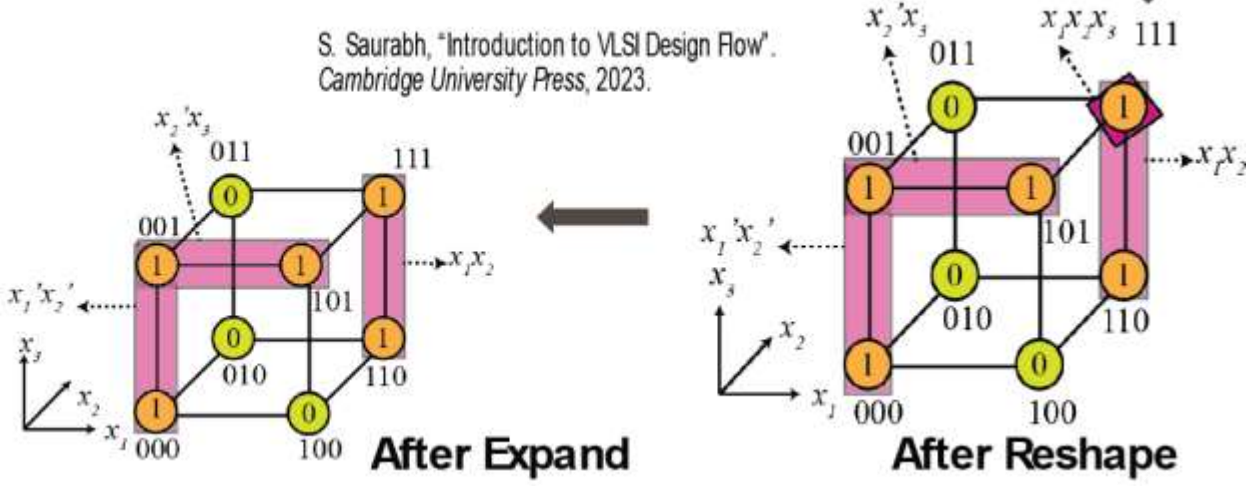
**Operators:**

- **Expand:** expands a non-prime implicant to make it prime (removes implicants covered by the expanded implicant)
- **Reduce:** replaces an implicant with a reduced implicant (covering fewer minterms) such that the function is still covered.
- **Reshape:** operates on a pair of implicants (expands one implicant and reduces others such that the function is still covered).
- **Irredundant:** makes a cover irredundant
  - Cover in which, if we remove any implicant, then it will be no more a cover.

# Heuristic Minimizer: Illustration




- Can carry irredundant operation at the end of iterations
- Not guaranteed to be minimum
- ESPRESSO two-level minimizer: practically very efficient




S. Saurabh, "Introduction to VLSI Design Flow". *Cambridge University Press*, 2023.

# References


- G. D. Micheli. “Synthesis and Optimization of Digital Circuits”. *McGraw-Hill Higher Education*, 1994.
- S. Saurabh, “Introduction to VLSI Design Flow”. Cambridge: *Cambridge University Press*, 2023.