Chapter 4
Optimized Implementation of Logic Functions

- Logic Minimization
- Karnaugh Maps
- Systematic Approach for Logic Minimization
- Minimization of Incompletely Specified Functions
- Tabular Method for Minimization
- Practical Considerations

Logic Minimization

- **Goal**: find the *optimum* implementation of logic function.
- The *cost criteria* for determining the optimum (*lowest cost*) implementation may be defined by the designer, for example:
  - Hardware cost, measured as total number & size of logic gates
  - Propagation delay (from inputs to outputs)
    - Worst case
    - Average
  - Design time (effort)
    - For example quick product prototypes may be required in initial phases of a project with optimizations applied later
- Equally optimal but different solutions are possible
- We consider two-level realizations of logic functions (eg. SOP or POS)
- Different logic minimization techniques:
  - Algebraic Manipulation
  - Karnaugh Maps (typically for functions of max 5 or 6 variables)
  - Tabular Method
  - Heuristic Techniques, eg. Espresso

Karnaugh Maps

x_1	x_2	f
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

(a) Truth table of 3-variable function

x_2 \ x_1	0	1
0	m_0	m_2
1	m_1	m_3

(b) Karnaugh map

x_1	x_2	x_3	f
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

(a) Truth table of 4-variable function

x_3 \ $x_1 x_2$	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

(b) Karnaugh map

Karnaugh Maps

Row number	x_1	x_2	x_3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

(a) The function $f(x_1, x_2, x_3) = \Sigma\, m(0, 2, 4, 5, 6)$.

x_3 \ x_1x_2	00	01	11	10
0	1	1	1	1
1	0	0	0	1

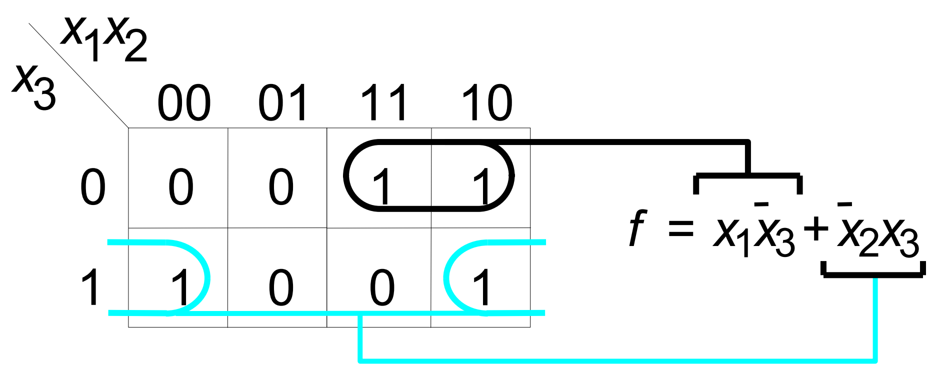
$$f = \bar{x}_3 + x_1\bar{x}_2$$

(b) SOP minimization of *f* using K-map

Karnaugh Maps

Row number	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

(a) The function $f(x_1, x_2, x_3) = \Sigma\, m(1, 4, 5, 6)$.



(b) SOP minimization of f using K-map

Karnaugh Maps

x_3x_4 \ x_1x_2	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

Figure 4.6. A four-variable Karnaugh map.

Karnaugh Maps

x_3x_4 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	0	0	1
10	1	0	0	1

$$f_1 = \bar{x}_2 x_3 + x_1 \bar{x}_3 x_4$$

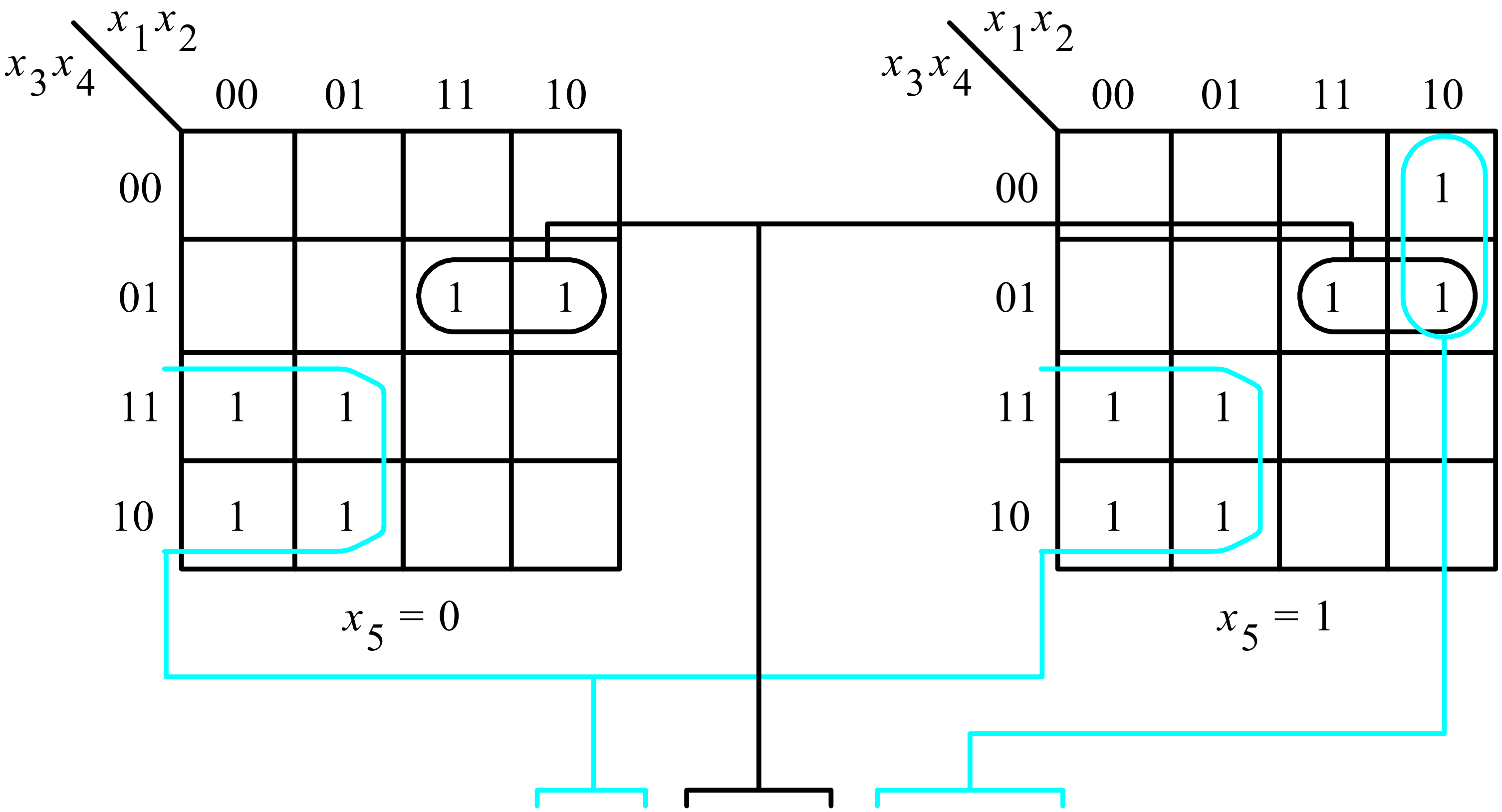
x_3x_4 \ x_1x_2	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	1	1	1
10	1	1	1	1

x_3x_4 \ x_1x_2	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	1	1	1	0
10	1	1	0	1

x_3x_4 \ x_1x_2	00	01	11	10
00	1	1	1	0
01	1	1	1	0
11	0	0	1	1
10	0	0	1	1

Figure 4.7. Examples of four-variable Karnaugh maps.

Karnaugh Maps



$$f_1 = \bar{x}_1x_3 + x_1\bar{x}_3x_4 + x_1\bar{x}_2\bar{x}_3x_5$$

Figure 4.8. A five-variable Karnaugh map.

Systematic Approach for Logic Minimization

- Intuitive strategy: find as few and as large as possible groups of 1s that cover all cases where the function has a value of 1.
- Each group of 1s represented by a single product term.
- The larger the group of 1s, the fewer the number of variables in the corresponding product term.
- To describe the systematic approach for logic minimization, precisely define terminologies:
- *Literal* - Each appearance of a variable in either normal or complemented form.
- *Implicant* – a product term *p* is said to be implicant of *f* iff for all input combination for which *p* evaluates to 1 *f* also evaluates to 1.
- The most basic implicants of *f* are its minterms.
- *Prime Implicant* – an implicant that can not be reduced (by combination) to another one with fewer literals.
- It is not possible to delete any literal in a prime implicant and still have a valid implicant.

Systematic Approach for Logic Minimization

- *Cover* – A collection of implicants that account for all valuations for which a given function is equal to 1 is called a cover of that function.
- Cover defines a particular implementation of the function. A number of different covers exist for a logic function. Example:
  - A set of all minters for which f = 1 is a cover.
  - A set of all prime implicants is a cover.
- *Cost* – the logic minimization criteria.
- For our purpose, we define the cost of a logic circuit as the number of gates plus the total number of inputs to all gates in the circuit.
- In general, the larger a circuit, the more important the cost issue becomes.
- Assumption: the primary input variables are available in both normal and complemented forms at zero cost. This is true, for example, in many PLDs.

Systematic Approach for Logic Minimization

- The lowest-cost implementation is achieved when the cover of a given function consists of prime implicants.
- How to determine the minimum-cost subset of prime implicants that will cover the function?
- If a prime implicant includes a minterm that is not included in any other prime implicant, such minterm is called *distinguished 1-cell*.
- A prime implicant containing one or more distinguished 1-cells is called *essential prime implicant*, and it must be included in the cover
- Steps of finding minimum-cost circuit:
  1. Generate all prime implicants for the given function *f*
  2. Find the set of essential prime implicants
  3. If the set of essential prime implicants covers all valuations for which *f* = 1, then this set is the desired cover of *f*. Otherwise, determine the non-essential prime implicants that should be added to form a complete minimum-cost cover.

Systematic Approach for Logic Minimization

- Example: Find the prime implicants, distinguished 1-cells, essential prime implicants and the minimum SOP representation for each of the following functions

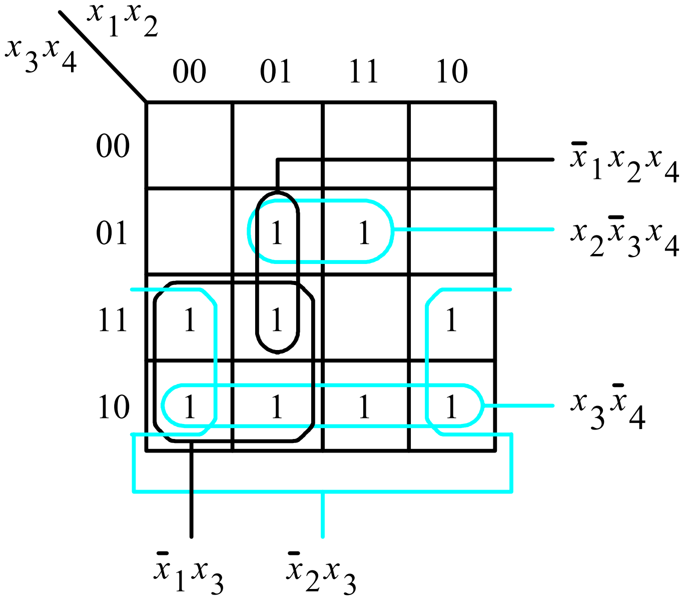


Figure 4.10. Four-variable function $f(x_1,\ldots,x_4) = \Sigma\, m(2, 3, 5, 6, 7, 10, 11, 13, 14)$.

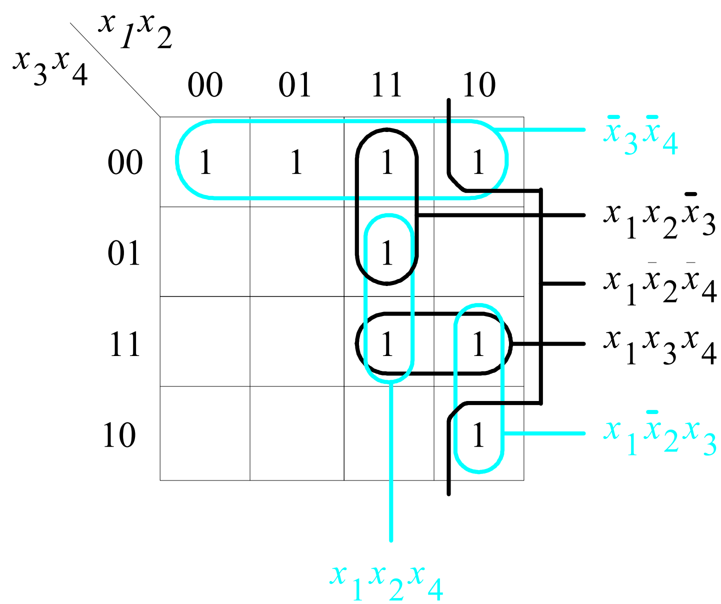


Figure 4.11. The function $f(x_1,\ldots,x_4) = \Sigma\, m(0, 4, 8, 10, 11, 12, 13, 15)$.

Systematic Approach for Logic Minimization

- Sometimes there may not be any essential prime implicants at all. Alternative solutions of equal weight possible.

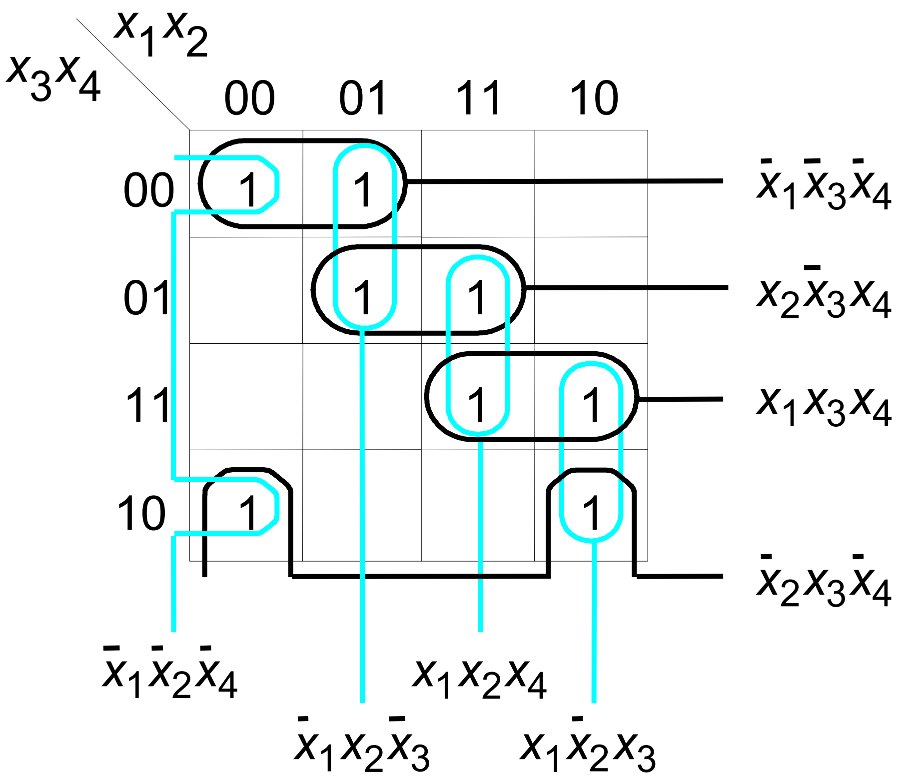


Figure 4.12. The function $f(x_1,\ldots,x_4) = \Sigma\, m(0, 2, 4, 5, 10, 11, 13, 15)$.

Systematic Approach for Logic Minimization

- *Minimization of POS*. Consider the maxterms for which f = 0 and combine them into sum terms that are of the largest groups possible. Then form the product of the minimum-cost cover of the sum terms.
- Alternatively, find the minimum-cost SOP of the complement of *f*, and then apply DeMorgan's theorem to the expression to obtain the minimum-cost POS realization of *f*.

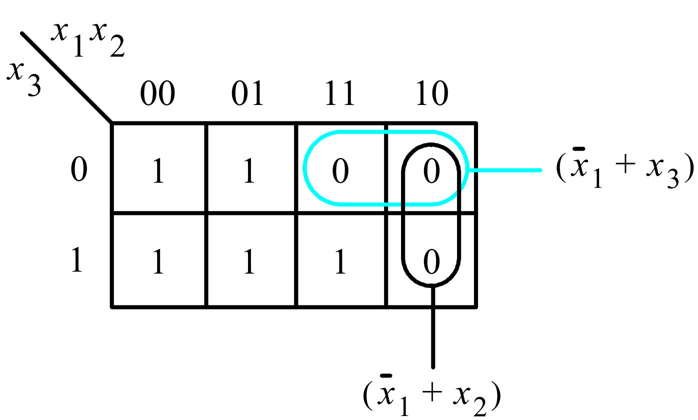


Figure 4.13. POS minimization of $f(x_1, x_2, x_3) = \Pi M(4, 5, 6)$.

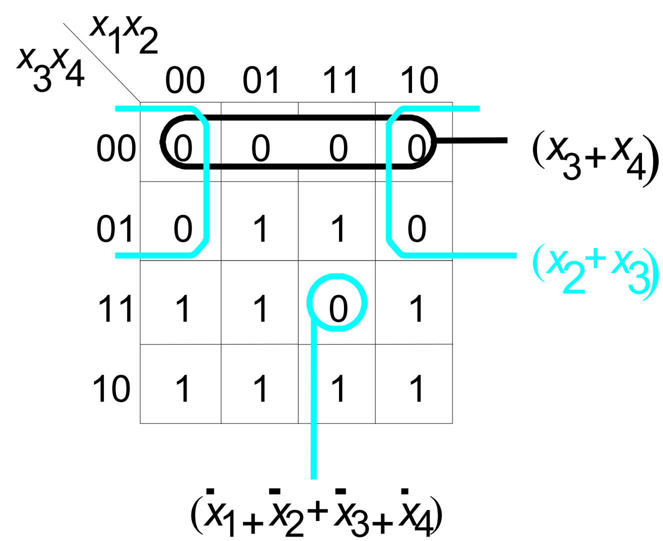
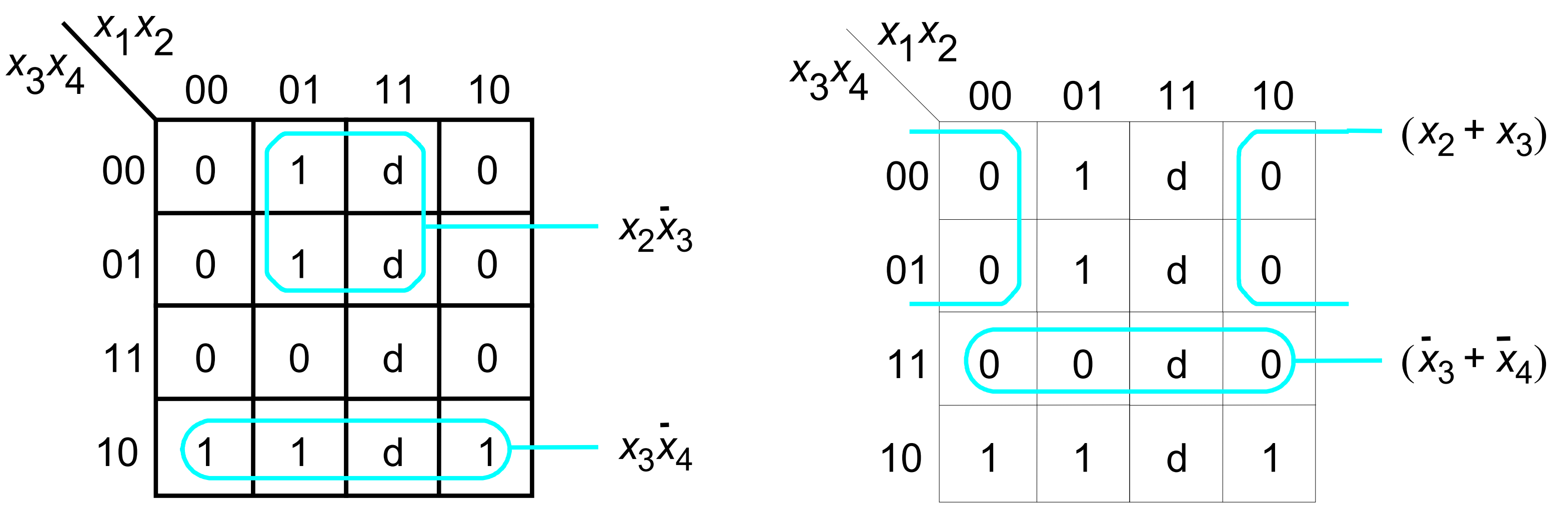


Figure 4.14. POS minimization of $f(x_1, \ldots, x_4) = \Pi M(0, 1, 4, 8, 9, 12, 15)$.

Minimization of Incompletely Specified Functions

- A function that has *don't care* conditions is said to be incompletely specified. This happens, for example, if certain input conditions can never occur.
- Don't cares can be used to advantage in the design of logic circuits.

Example: Find the simplest SOP and POS for the following function.



(a) SOP implementation

(b) POS implementation

Figure 4.15. Two implementations of the function $f(x_1,\ldots,x_4) = \Sigma\, m(2, 4, 5, 6, 10) + D(12, 13, 14, 15)$.

Tabular Method for Minimization

- The technique described here is called *Quine-McClusky methd*.
- We will use cube representation for the prime implicants.
- This method involves two major tasks:
  1) Generation of all prime implicants of the logic function
  2) Determination of a minimum-cost cover from all the prime implicants
- The steps are detailed below:

1. Generate all the prime implicants by successive pairwise comparison of the cubes
2. Derive a cover table which indicates the minterms of *f* covered by each prime implicant
3. Include the essential prime implicants (if any) in the final cover and reduce the table by removing both these prime implicants and the covered minterms
4. Use the concept of row and column dominance to reduce the cover table further. A dominated row is removed only if the cost of its prime implicant is greater than or equal to the cost of the dominating row's prime implicant.
5. Repeat steps 3 and 4 until the cover table is either empty or no further reduction is possible
6. If the reduced cover table is not empty, then use the branching approach to determine the remaining prime implicants that should be included in a minimum-cost cover

Tabular Method for Minimization

Example: Determine all the prime implicants of the function defined by

$$f\,(x_1,\ldots,\, x_4) = \Sigma m(0, 4, 8, 10, 11, 12, 13, 15)$$

- Next find a minimum-cost cover for *f*.

Final minimal cover of f is given by: C =

Thus, the minimum-cost SOP for f is: f =

Tabular Method for Minimization

- The tabular method can also be used for minimization of functions with don’t care conditions.
- Example: $f(x_1,\ldots, x_4) = \Sigma m(0, 2, 5, 6, 7, 8, 9, 13) + D(1, 12, 15)$

Final minimal cover of f is given by: C =

Thus, the minimum-cost SOP for f is: f =

Tabular Method for Minimization

- In the following example, there are no essential prime implicants and no dominant rows or colums. Moreover, all prime implicants have the same cost. In this case the concept of *branching* is used.

  $f(x_1,\ldots, x_4) = \Sigma m(0, 3, 10, 15) + D(1, 2, 7, 8, 11, 14)$

Practical Considerations

- The Tabular method is relatively easy to understand but has some drawbacks if considered for CAD implementation.
- The main difficulty is that the number of cubes that must be considered in the process can be extremely large as it first tries to obtain *all* the prime implicants of the function.
- Heuristic techniques that produce good results in reasonable time are preferred specially for functions of large number of variables.
- The *Espresso* algorithm from UC Berkley is one such example.
- CAD tools provide logic synthesis software that can be used to target various types of chips, such as PLDs, gate arrays, standard cells, and custom chips.
- The process in logic design & implementation by such tools include:
  - *Technology-independent logic synthesis*
  - *Technology mapping*