CS1026 – Digital Logic Design

Boolean Algebra III

Shane Sheehan [1]

[1]ADAPT
Trinity College Dublin

January 16, 2017

Today's Overview

But what about Max-terms

Min-terms and Max-terms are complements of each other:

- $M_i' = m_i$
- $M_i = m_i'$

Hint: Use DeMorgan's Theorem to prove this

But why?

Minimisation of literals and operators:

- Reduces the number of gates to implement the function
- Reduces the cost of implementation

Systematic Algebraic Reduction (SAR)

- Uses algebraic theorems and postulates!

The SAR algorithm I

Step 1:

- Expand the function into its Standard sum of products (SOP)

Hint

This includes:

- All variables
- In order to it easier to recognise patterns

The SAR algorithm II

Step 2:

- Compare all pairs of products for:
  1. Adjacency Theorem – $A.B + A.B' = A$
  2. Idempotency Theorem – $A + A = A$

Hint

You might need to repeat this multiple times!

The SAR algorithm III

Step 3:

- Once you have done all reductions possible in step 2,
  - See if the Consensus Theorem applies

Consensus Theorem

$$A.B + A'.C + B.C = A.B + A'.C$$

SAR Algorithm Example I

Using SAR minimise the function:

- $F = (A + B + C).(A' + C).(B' + C)$

SAR Algorithm Example II

Step 1:

- Use truth table to derive the min-terms

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

SAR Algorithm Example III

Step 2:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- Write F in compact Min-term form
- $F = A'.B'.C + A'.B.C + A.B'.C + A.B.C$

SAR Algorithm Example IV

Step 3:

- Apply Adjacency Theorem to *all possible* pairs
- In: $F = A'.B'.C + A'.B.C + A.B'.C + A.B.C$
  - Terms 1 & 2 $\implies A'.B'.C + A'.B.C = A'.C$
  - Terms 1 & 3 $\implies A'.B'.C + A.B'.C = B'.C$
  - Terms 1 & 4 $\implies$ N/A
  - Terms 2 & 3 $\implies$ N/A
  - Terms 2 & 4 $\implies A'.B.C + A.B.C = B.C$
  - Terms 3 & 4 $\implies A.B'.C + A.B.C = A.C$
- $\therefore F = A'.C + B'.C + B.C + A.C$

SAR Algorithm Example V

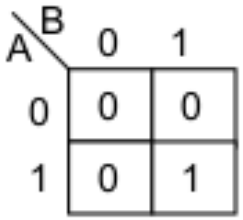
Step 4:

- Do a second pass with the Adjacency theorem
  - $F = A'.C + B'.C + B.C + A.C$
  - $\therefore F = C + C = C$

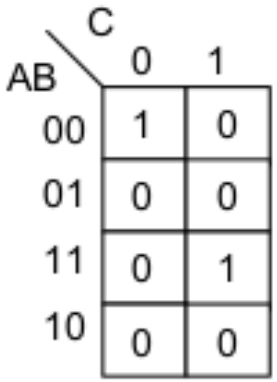
Beware!

- No need to apply Consensus Theorem
- We cannot simplify further

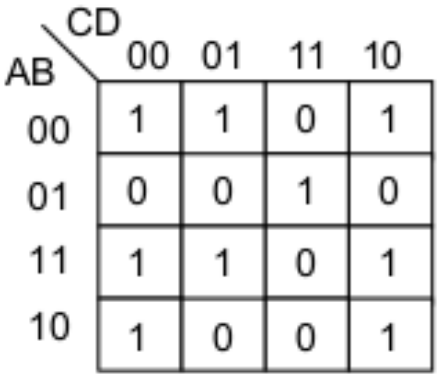
SAR for machines.. K-Maps for humans



F(A,B)
2-Variables



F(A,B,C)
3-Variables



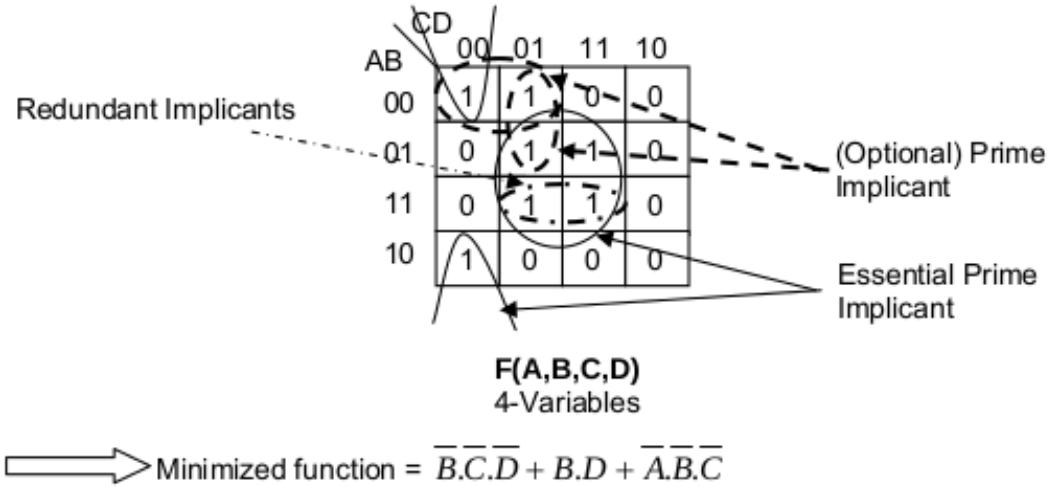
F(A,B,C,D)
4-Variables

For minimisation of five or fewer variables

- Humans use a K-Map
- Your brain can do this really well

More K-Map definitions I

Let's start with an example:



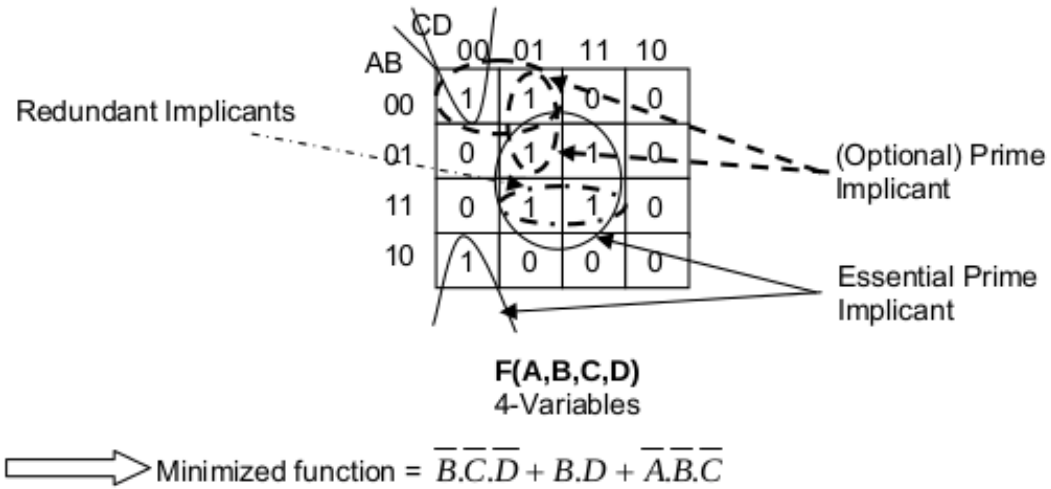
More K-Map definitions II

Implicant

- The product term where the function is evaluated to 1 or complemented to 0
- Implies the term of the function is 1 (or complemented to 0)
- Thus each square with a 1 for the function denotes an implicant (p)
- If the complement of the function is being discussed, then 0s denote implicants (r)

Note: To find the complement of F, apply the same rules to 0 entries in the K-Map instead of 1

More K-Map definitions III

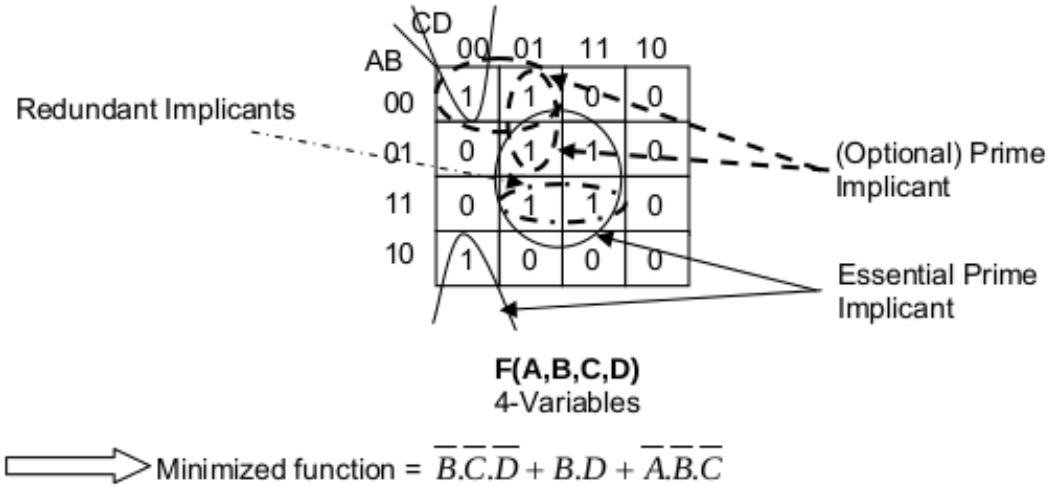


Prime Implicant

- Rectangular group of product terms

You should see them contained in a single larger implicant!

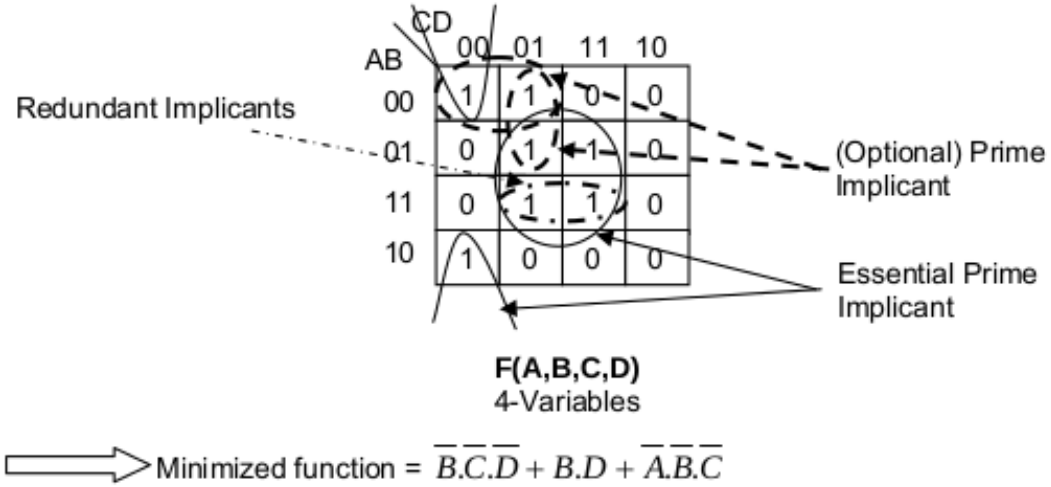
More K-Map definitions IV



Essential Prime Implicant

- Provides the only coverage for a given min-term

More K-Map definitions V

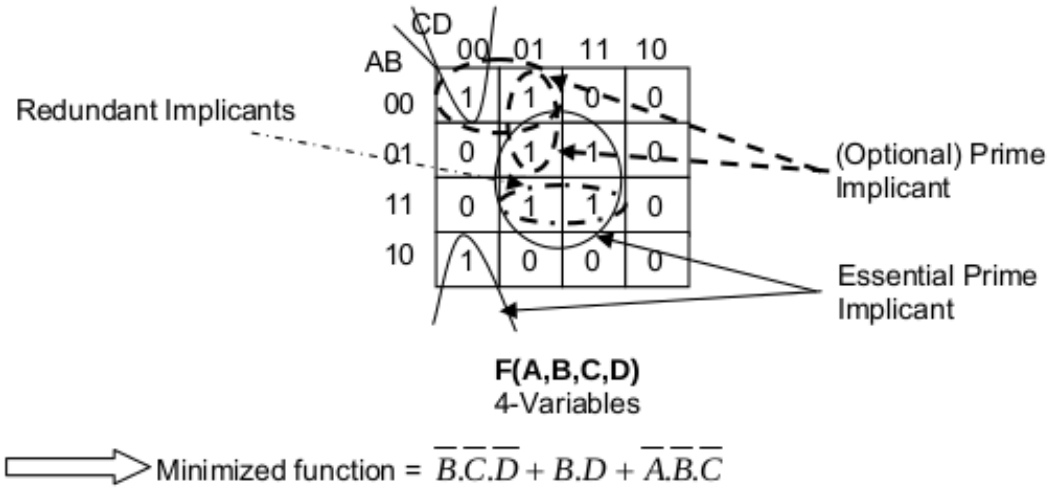


Optional Prime Implicant

- Provides an alternate covering for a given Min-term

Note: We can write some functions in a minimum form using more than one way because of optional prime implicants

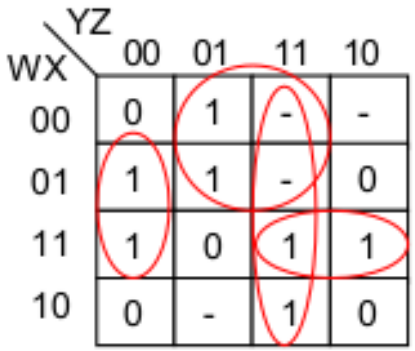
More K-Map definitions VI



Redundant Prime Implicant

- Product term that represents a square completely covered by other essential or optional prime implicants

But what if I Don't Care?! [Damiani and De Micheli, 1993]



$F(W, X, Y, Z)$ with unspecified values (dont cares, "$-$"):

- $F(W, X, Y, Z) = X.Y'.Z' + W'.Z + Y.Z + W.X.Y$

Representing "dont care" min-terms in compact form:

- $\sum md(1, 4, 5)$ – Note: *md* refers to *dont care* min-terms

References (Homework) I


Damiani, M. and De Micheli, G. (1993).
Don't care set specifications in combinational and synchronous logic circuits.
Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 12(3):365–388.