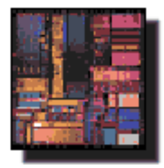
Digital Systems Design Automation

Unit 2: Advanced Boolean Algebra

Lecture 2.5: Co-factors of Boolean Functions



Anand Raghunathan
raghunathan@purdue.edu

Outline

2.1 Boolean algebra: Quick review
2.2 Boolean spaces and functions
2.3 Boolean function representations
2.4 Conversion of Boolean function representations
2.5 Co-factors of Boolean functions
2.6 Boolean difference and Quantification

Operations on Boolean functions

- The usual suspects…
  - Complement
  - AND
  - OR
  - XOR
  - …
- Co-factoring: A new operation on Boolean functions
- Applications of co-factoring
  - Shannon's expansion
  - Boolean difference
  - Existential and Universal quantification

Co-factors of Boolean Functions

- A co-factor of a function is derived by fixing one of the input variables to a constant (0 or 1), resulting in a new function of n-1 variables
- Given a function $f(x_1 \ldots x_n)$
  - Positive co-factor w.r.t. x_i is defined as
    $f_{x_i}(x_1 \ldots x_{i-1}, x_{i+1} \ldots x_n) = f(x_1 \ldots x_{i-1}, \mathbf{x_i = 1}, x_{i+1} \ldots x_n)$
  - Negative co-factor w.r.t. x_i is defined as
    $f_{x_i'}(x_1 \ldots x_{i-1}, x_{i+1} \ldots x_n) = f(x_1 \ldots x_{i-1}, \mathbf{x_i = 0}, x_{i+1} \ldots x_n)$

Examples

$f = ab + bc + ac$

$f_a = 1.b + bc + 1.c = b + c$

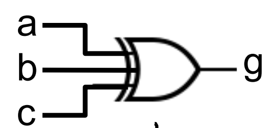
$f_{a'} = bc$

$f_b = a + c$

$f_{b'} = ac$

$f_c = a + b$

$f_{c'} = ab$



$g_a = (b \oplus c)' = bc + b'c'$

$g_{a'} = b \oplus c = b'c + bc'$

$g_b =$

$g_{b'} =$

$g_c =$

$g_{c'} =$

Co-factors of Boolean Functions

- Also called
  - Shannon co-factors
  - Restriction of a function on a variable
- Can be applied on multiple variables

  $f_{x_i x_j'} = f(x_1 \ldots x_i = 1 \ldots x_j = 0 \ldots x_n)$
- Order does not matter

  $f_{x_i x_j} = (f_{x_i})_{x_j} = (f_{x_j})_{x_i}$
- Co-factor w.r.t. a cube

Examples:

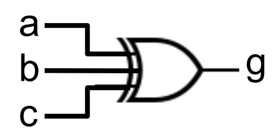
$f = ab + bc + ac$

$f_{ab} = 1$

$f_{ab'} = c$

$f_{a'b'c'} = 0$

$f_{ab'c} = 1$



$g_{ab} = 1 \oplus 1 \oplus c = c$

$g_{a'b} = c'$

$g_{b'c'} = a$

$g_{abc'} = 0$

Properties of Co-factors

- Given two functions f(x) and g(x)
- How can we compute co-factors of a function h that is derived from f and g?

Function	Co-factors
$h(x) = f'(x)$	$h_{x_i} = (f_{x_i})'$ $h_{x_i'} = (f_{x_i'})'$
$h(x) = f(x)$ AND $g(x)$	$h_{x_i} = f_{x_i}$ AND g_{x_i} $h_{x_i'} = f_{x_i'}$ AND $g_{x_i'}$
$h(x) = f(x)$ OR $g(x)$	$h_{x_i} = f_{x_i}$ OR g_{x_i} $h_{x_i'} = f_{x_i'}$ OR $g_{x_i'}$
$h(x) = f(x)$ XOR $g(x)$	$h_{x_i} = f_{x_i}$ XOR g_{x_i} $h_{x_i'} = f_{x_i'}$ XOR $g_{x_i'}$

Co-factor of complement is complement of co-factor

Co-factor of AND is AND of co-factors

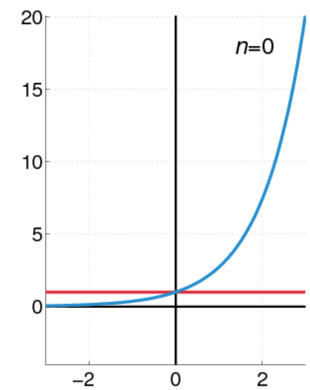
Co-factor of OR is OR of co-factors

Co-factor of XOR is XOR of co-factors

The co-factor operation distributes over any binary operator

OK, so why do we need Co-factors?

- Many applications… for example:
- Recall **Taylor series** from high-school math?
  - A representation of a (real or complex) function as a sum of polynomial terms (1, x, x^2, x^3, x^4, …)
    - Example: $e^x = 1 + x + x^2/2! + x^3/3! + \ldots$
- Question: Is there a similar concept for Boolean functions?
  - Can we express a Boolean function in terms of "simpler" functions?



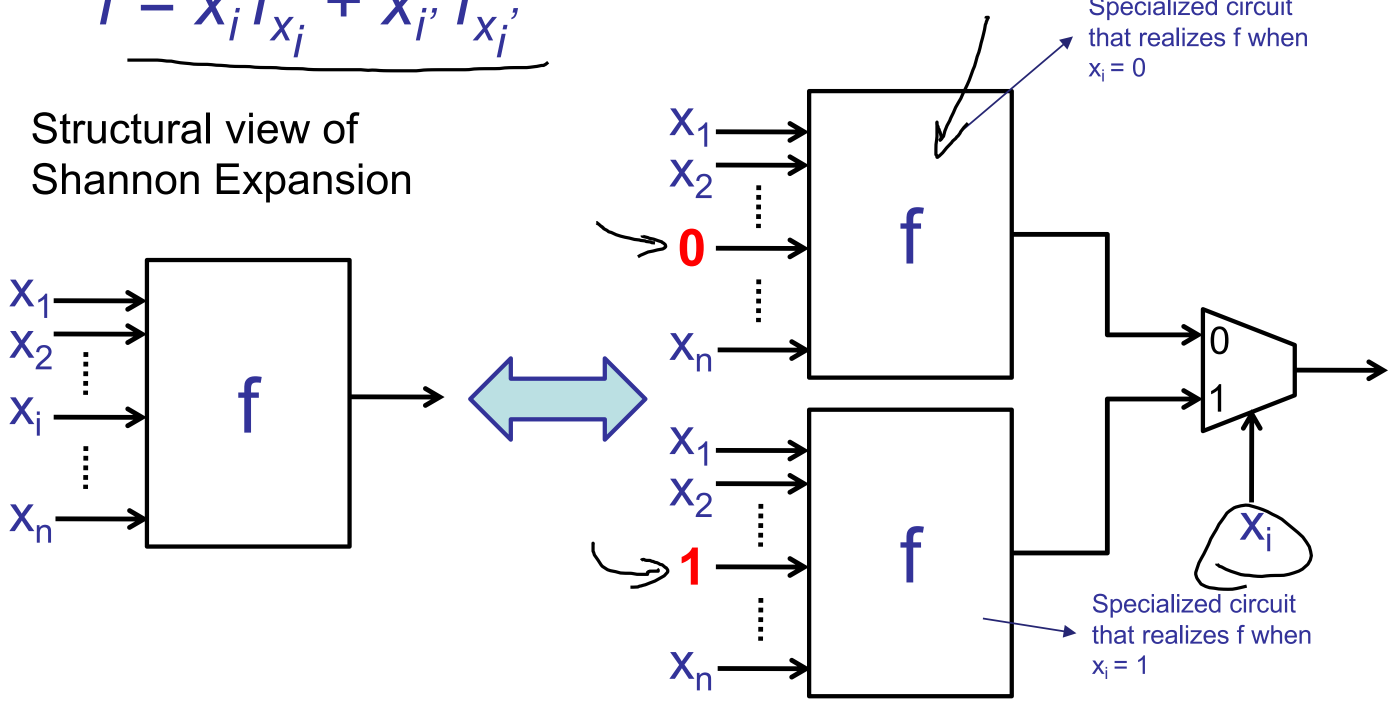
Animation of Taylor series for e^x (Source: Wikipedia)

Shannon's (Boole's) Expansion Theorem

- Given a Boolean function $f(x_1 \dots x_n)$ and any variable x_i

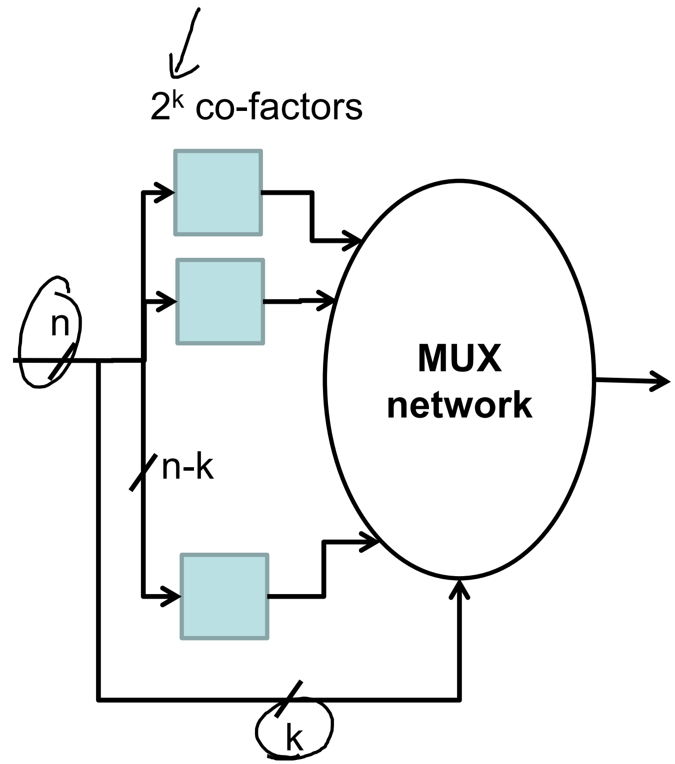
$$f = x_i f_{x_i} + x_{i'} f_{x_i'}$$

Structural view of Shannon Expansion



Shannon Expansion

- Can be applied recursively to "decompose" a function into simpler functions
  - In the extreme case, just a network of multiplexers
- Also called **Shannon Decomposition**



Shannon Expansion

- Example

  $f = xy + zw' + x'w'$

$f_w = xy \quad f_{w'} = xy + z + x' = y + z + x'$

$f = w' f_{w'} + w f_w = w'(y + z + x') + w(xy)$

$= w'[x' \cdot 1 + x(y+z)] + w[x' \cdot 0 + x \cdot y]$