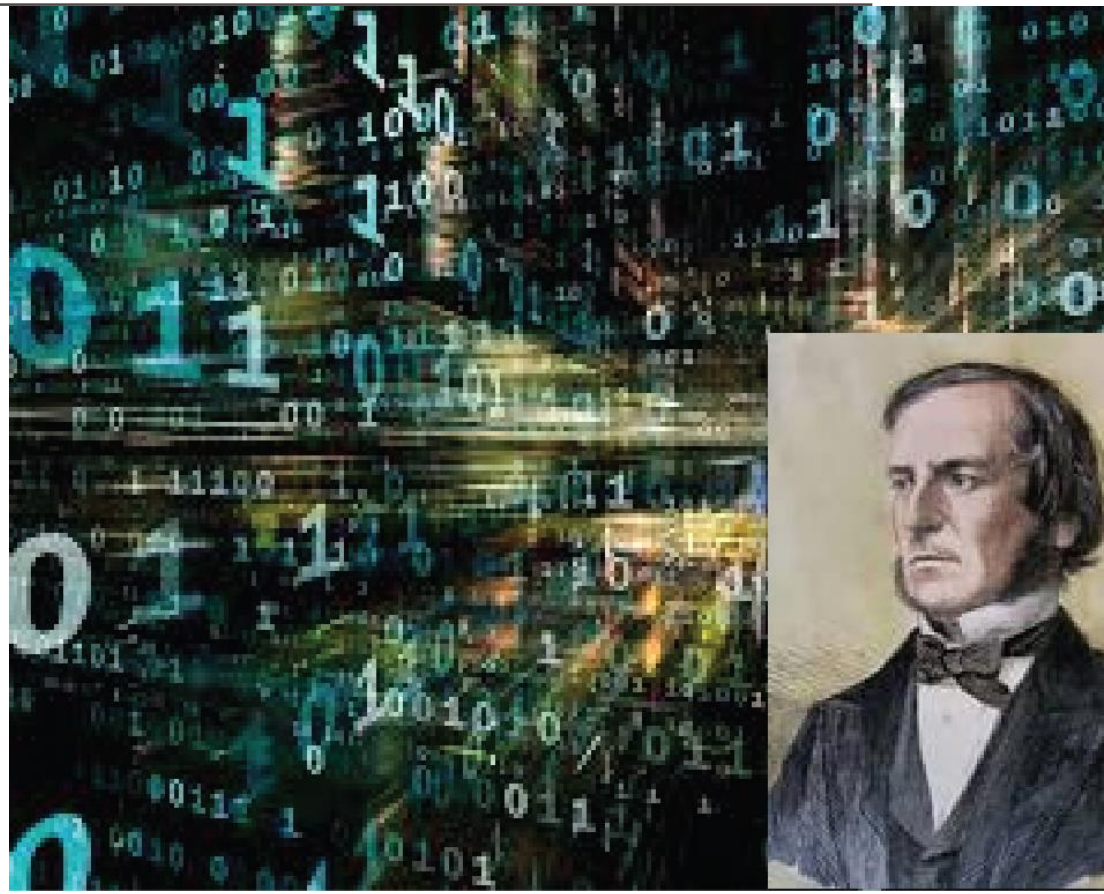
DIGITAL CIRCUITS

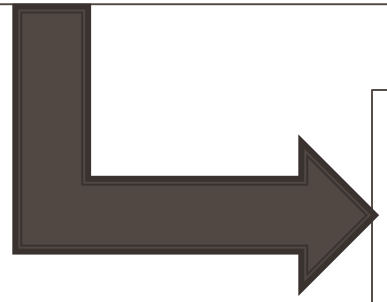


Week-8, Lecture-1
Multiplexers

Sneh Saurabh
25th September, 2018

Digital Circuits: Announcements/Revision

Combinational Circuit Design



Using Multiplexers

Shannon's Expansion Theorem

Shannon's Expansion Theorem:

Any Boolean function $f(w_1, w_2, \dots, w_n)$ can be written in the form:

$$f(w_1, w_2, \dots, w_n) = w_1'.f(0, w_2, \dots, w_n) + w_1.f(1, w_2, \dots, w_n)$$

- The term $\boldsymbol{f(0, w_2, \dots, w_n)}$ is known as the cofactor of f with respect to w_1' (also called ***negative cofactor*** with respect to w_1)
- It is obtained by replacing all occurrences of w_1 in f with 0
- It is denoted as $f_{w_1'}$

- The term $\boldsymbol{f(1, w_2, \dots, w_n)}$ is known as the cofactor of f with respect to w_1 (also called ***positive cofactor*** with respect to w_1)
- It is obtained by replacing all occurrences of w_1 in f with 1
- It is denoted as f_{w_1}

$$f(w_1, w_2, \dots, w_n) = w_1'.f_{w_1'} + w_1.f_{w_1}$$

Shannon's Expansion Theorem: Illustration-1

Problem:

Do Shannon's expansion of: $f = w_1 \oplus w_2$ and implement in terms of 2-to-1 MUX.

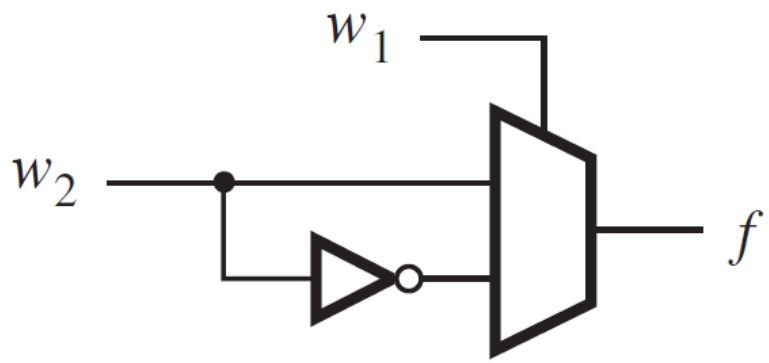
Solution:

$$f = w_1 \oplus w_2 = w_1'.w_2 + w_1 w_2'$$

$$f_{w_1'} = w_2$$

$$f_{w_1} = w_2{}'$$

$$f = w_1'.f_{w_1'} + w_1.f_{w_1}$$



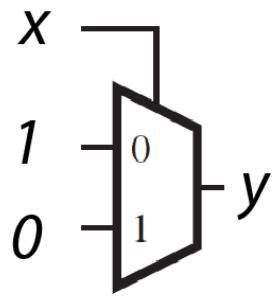
Shannon's Expansion Theorem: Applied Recursively

- Shannon Expansion Theorem can be applied to the co-factors (with respect to variables present in the co-factors)

- The expansion can be done again and again to the co-factors (recursively)
  - ➢ The recursion can be stopped when co-factors consist of only one variable or constants (“0” or “1”)
- Shannon Expansion when applied recursively will yield a 2-to-1 MUX implementation of a function.

- The number of 2-to-1 MUX can be reduced by sharing MUXES that realize the same function

- Complement of a variable can be realized using one 2-to-1 MUX



Shannon's Expansion Theorem: Applied Recursively

Problem:

Implement the function $f = w_1w_2 + w_1w_3 + w_2w_3$ using only **2-to-1 MUX**.

Solution:

Step 1: Expand with respect to w_1

$$f = w_1'.f_{w_1'} + w_1.f_{w_1}$$

$$f_{w_1'} = \boldsymbol{w_2w_3}$$

$$f_{w_1} = \boldsymbol{w_2 + w_3 + w_2w_3}$$

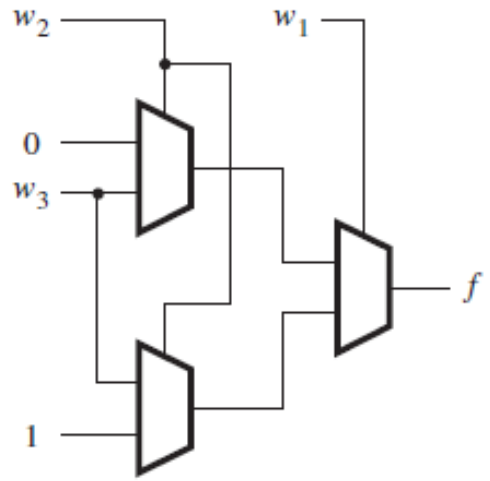
Step 2: Expand each co-factors with respect to w_2

$$f_{w_1'} = w_2'.f_{w_1'w_2'} + w_2.f_{w_1'w_2}$$

$$f_{w_1'w_2'} = \boldsymbol{0} \qquad f_{w_1'w_2} = \boldsymbol{w_3}$$

$$f_{w_1} = w_2'.f_{w_1w_2'} + w_2.f_{w_1w_2}$$

$$f_{w_1w_2'} = \boldsymbol{w_3} \qquad f_{w_1w_2} = \boldsymbol{1}$$



Shannon's Expansion Theorem: Applied Recursively

Problem:

Implement the function $f = w_1w_2 + w_1w_3 + w_2w_3$ using only **4-to-1 MUX**.

Solution:

Step 1: Expand with respect to w_1

$f = w_1'.f_{w_1'} + w_1.f_{w_1}$

$f_{w_1'} = \boldsymbol{w_2w_3}$

$f_{w_1} = \boldsymbol{w_2 + w_3 + w_2w_3}$

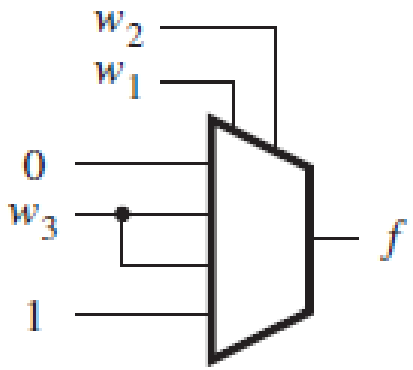
Step 2: Expand each co-factor with respect to w_2

$f_{w_1'} = w_2'.f_{w_1'w_2'} + w_2.f_{w_1'w_2}$

$f_{w_1'w_2'} = \boldsymbol{0}$ $\quad f_{w_1'w_2} = \boldsymbol{w_3}$

$f_{w_1} = w_2'.f_{w_1w_2'} + w_2.f_{w_1w_2}$

$f_{w_1w_2'} = \boldsymbol{w_3}$ $\quad f_{w_1w_2} = \boldsymbol{1}$



Shannon's Expansion Theorem: Significance of Variables

- Choosing different variables for expansion can yield different implementations.
- If the order of variables in which the function is expanded is changed, then different implementations obtained can have different costs

Problem: Compute the cost of the implementation of the function $\boldsymbol{f = w_1'w_3 + w_2w_3'}$ using only 2-to-1 MUX, when the function is expanded with respect to i) w_1 ii) w_2 iii) w_3

Solution:

i) Expand with respect to w_1

$$f = w_1'.f_{w_1'} + w_1.f_{w_1}$$

$$f_{w_1'} = w_3 + w_2w_3' = (w_3 + w_2)(w_3 + w_3')$$

$$= \boldsymbol{w_3 + w_2}$$

$$f_{w_1} = \boldsymbol{w_2w_3'}$$

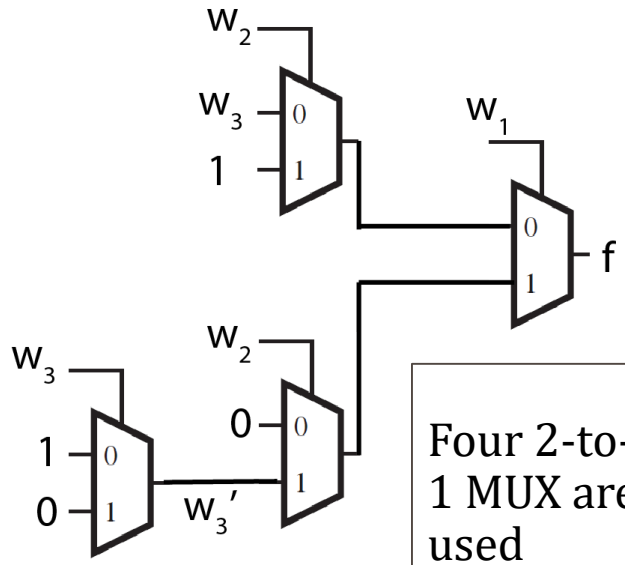
Step 2: Expand each co-factor with respect to w_2

$$f_{w_1'} = w_2'.f_{w_1'w_2'} + w_2.f_{w_1'w_2}$$

$$f_{w_1'w_2'} = \boldsymbol{w_3} \qquad f_{w_1'w_2} = \boldsymbol{1}$$

$$f_{w_1} = w_2'.f_{w_1w_2'} + w_2.f_{w_1w_2}$$

$$f_{w_1w_2'} = \boldsymbol{0} \qquad f_{w_1w_2} = \boldsymbol{w_3'}$$



Four 2-to-1 MUX are used

Shannon's Expansion Theorem: Significance of Variables

Problem: Compute the cost of the implementation of the function $f = w_1'w_3 + w_2w_3'$ using only 2-to-1 MUX, when the function is expanded with respect to i) w_1 ii) w_2 iii) w_3

Solution:

i) Expand with respect to w_1

$f = w_1'.f_{w_1'} + w_1.f_{w_1}$

$f_{w_1'} = w_3 + w_2w_3' = (w_3 + w_2)(w_3 + w_3')$

$= \boldsymbol{w_3 + w_2}$

$f_{w_1} = \boldsymbol{w_2w_3'}$

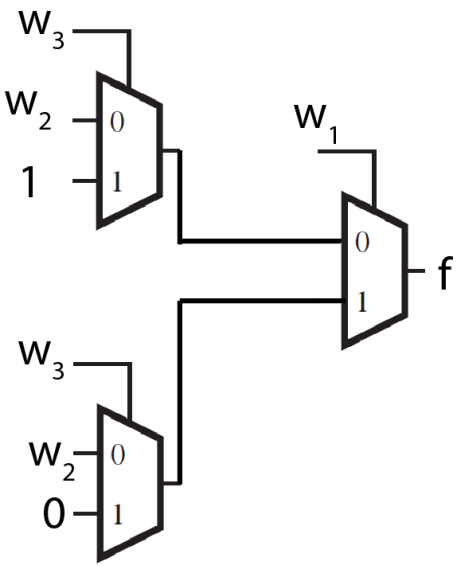
Step 2: Expand each co-factor ***with respect to*** $\boldsymbol{w_3}$ *instead of* w_2

$f_{w_1'} = w_3'.f_{w_1'w_3'} + w_3.f_{w_1'w_3}$

$f_{w_1'w_3'} = \boldsymbol{w_2}$ $\quad f_{w_1'w_3} = \boldsymbol{1}$

$f_{w_1} = w_3'.f_{w_1w_3'} + w_3.f_{w_1w_3}$

$f_{w_1w_3'} = \boldsymbol{w_2}$ $\quad f_{w_1w_3} = \boldsymbol{0}$



Three 2-to-1 MUX are used instead of four

Shannon's Expansion Theorem: Significance of Variables

Problem: Compute the cost of the implementation of the function $f = w_1'w_3 + w_2w_3'$ using only 2-to-1 MUX, when the function is expanded with respect to i) w_1 ii) w_2 iii) w_3

Solution:

ii) Expand with respect to w_2

Similar implementation. Cost is four 2-to-1 MUXes.

Do it yourself…

Shannon's Expansion Theorem: Significance of Variables

Problem: Compute the cost of the implementation of the function $f = w_1'w_3 + w_2w_3'$ using only 2-to-1 MUX, when the function is expanded with respect to i) w_1 ii) w_2 iii) w_3

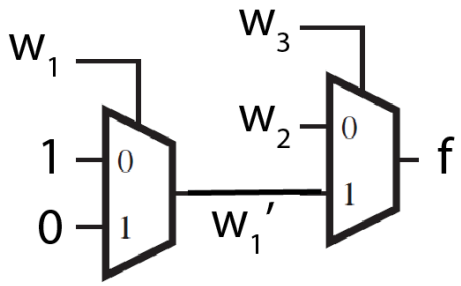
Solution:

iii) Expand with respect to w_3

$f = w_3'.f_{w_3'} + w_3.f_{w_3}$

$f_{w_3'} = \boldsymbol{w_2}$

$f_{w_3} = \boldsymbol{w_1'}$



Two 2-to-1 MUX are sufficient

Conclusions:

- The order of variables chosen in the recursive Shannon's Expansion has a big impact on the cost of implementation
- In practice, tools try out various orders and come up with a minimized implementation

Implementing Boolean Functions using Multiplexers

Problem 1: Implement the function $f(x, y, z) = \Sigma m(3,5,6,7)$ using minimum 2-to-1 MUX (do not use any other gates)

Solution:

i) Draw the K-map

x \ yz	00	01	11	10
0	0	0	1	0
1	0	1	1	1

ii) Take x as the select input of the MUX:

$$f(x, y, z) = x'(yz) + x(y + z)$$

Can be implemented using three 2-to-1 MUX

iii) Take y as the select input of the MUX:

$$f(x, y, z) = y'(xz) + y(x + z)$$

Can be implemented using three 2-to-1 MUX

iv) Take z as the select input of the MUX:

$$f(x, y, z) = z'(xy) + z(x + y)$$

Can be implemented using three 2-to-1 MUX

Implementing Boolean Functions using Multiplexers

Problem 2: Implement the function $f(w, x, y, z) = \Sigma m(1,4,5,7,9,12,13)$ using a 4-to-1 MUX and minimum number of 2-to-1 MUX only.

Solution:

i) Draw the K-map

wx \ yz	00	01	11	10
00	0	1	0	0
01	1	1	1	0
11	1	1	0	0
10	0	1	0	0

- For the 4-to-1 MUX, two select signals must be chosen out of four
- There can be six different combinations: wx, xy, yz, wz, wy, xz
- Different combination can require different number of 2-to-1 MUX
- The combination with minimum number of 2-to-1 MUX must be chosen

Implementing Boolean Functions using Multiplexers

Problem 2: Implement the function $f(w,x,y,z) = \Sigma m(1,4,5,7,9,12,13)$ using a 4-to-1 MUX and minimum number of 2-to-1 MUX only.

Solution:

i) Draw the K-map

wx \ yz	00	01	11	10
00	0	1	0	0
01	1	1	1	0
11	1	1	0	0
10	0	1	0	0

For wx:

$00: y'z$

$01: y' + z$

$10: y'z$

$11: y'$

For xy:

$00: z$

$01: 0$

$10: 1$

$11: w'z$

For yz:

$00: x$

$01: 1$

$10: 0$

$11: w'x$

For wz:

$00: xy'$

$01: x + y'$

$10: xy'$

$11: y'$

For wy:

$00: x + z$

$01: xz$

$10: 0$

$11: x + z$

For xz:

$00: 0$

$01: y'$

$10: y'$

$11: y' + w'$

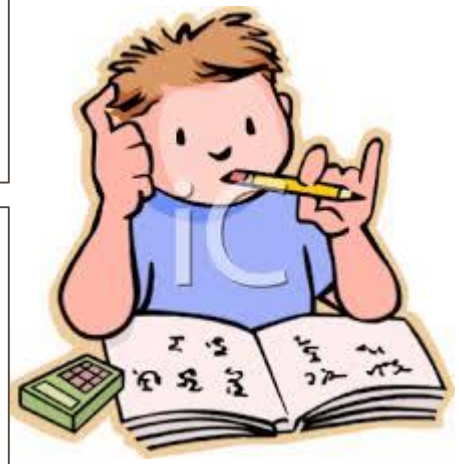
Digital Circuits: Practice Problems

Problems 4.31-4.35

from “Digital Design”– M. Morris Mano & Michael D. Ciletti, Ed-5, Pearson (Prentice-Hall).

Problems 6.3-6.10, 4.9, 4.32

from Fundamentals of Digital Logic with Verilog Design - S. Brown, Z. Vranesic



Problem: Design XNOR gate using 4:1 multiplexer

Problem: Design XNOR gate using 2:1 multiplexers

Problem: Implement the function $f(w,x,y,z) = \Sigma m(1,4,5,7,9,12,13)$ using a 8-to-1 MUX and minimum number of 2-to-1 MUX only.

Problem: Implement the function $f(w,x,y,z) = \Sigma m(1,4,5,7,9,12,13)$ using a 8-to-1 MUX and minimum number of logic-gates only.