Logic Design Styles

Dinesh Sharma

Microelectronics Group, EE Department
IIT Bombay, Mumbai

July 26, 2016

Complementary Pass gate Logic

- This logic family is based on multiplexer logic.
- Given a Boolean function $F(x_1, x_2, \ldots, x_n)$, we can express it as:

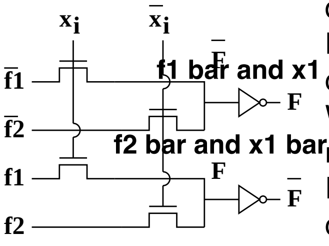
  $$F(x_1, x_2, \ldots, x_n) = x_i \cdot f1 + \overline{x_i} \cdot f2$$

  where f1 and f2 are reduced expressions for F with x_i forced to 1 and 0 respectively.
- Thus, F can be implemented with a multiplexer controlled by x_i which selects f1 or f2 depending on x_i.
- f1 and f2 can themselves be decomposed into simpler expressions by the same technique.

Complementary Pass gate Logic

- To implement a multiplexer, we need both x_i and $\overline{x_i}$.
- Therefore, this logic family needs all inputs in true as well as in complement form.
- In order to drive other gates of the same type, it must produce the outputs also in true and complement forms.
- Thus each signal is carried by two wires.
- This logic style is called “Complementary Pass-gate Logic” or CPL for short.

Basic Multiplexer Structure



Pure pass-gate logic contains no ‘amplifying’ elements. Therefore, each logic stage degrades the logic level.

Hence, multiple logic stages cannot be cascaded.

We include conventional CMOS inverters to restore the logic level.

Ideally, the multiplexer should be composed of complementary pass gate transistors.

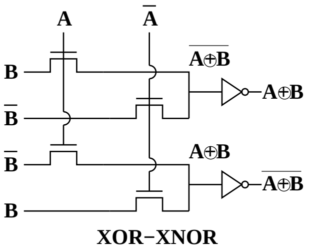
However, we shall use just n channel transistors as switches for simplicity.

Logic Design using CPL

- For any logic function, we pick one input as the control variable.
- Multiplexer inputs are decided by re-evaluating the function, forcing this variable to 1 and zero respectively.
- Since both true and complement outputs are generated by CPL, we need fewer types of gates.
- For example, we do not need separate gates for AND and NAND functions.
- The same applies to OR-NOR, and XOR-XNOR functions.

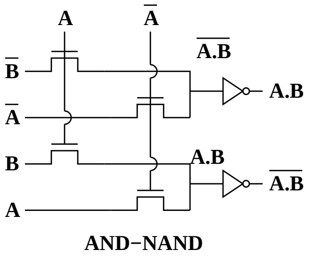
Implementation of XOR and XNOR

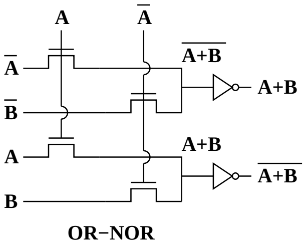
To take an example, let us consider the XOR-XNOR functions.



- Because of the inverter, for XOR output, We calculate the XNOR function given by $A.B + \overline{A}.\overline{B}$.
- If we put A = 1, this reduces to B and for A = 0, it reduces to $\overline{B}$.
- For the XNOR output, we generate the XOR expression = $A.\overline{B} + \overline{A}.B$
- The expression reduces to $\overline{B}$ for A = 1 and to B for A = 0.

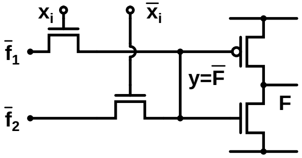
Implementation of AND-NAND and OR-NOR

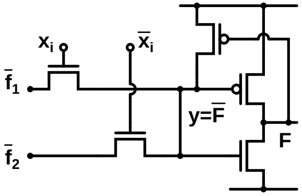




- For AND, the mux should output $\overline{A.B}$ to be inverted by the buffer. This reduces to $\overline{B}$ when A = 1 and to 1 (= $\overline{A}$) when A = 0.
- Implementation of NAND, OR and NOR functions follows along the same lines.

Buffer Leakage Current



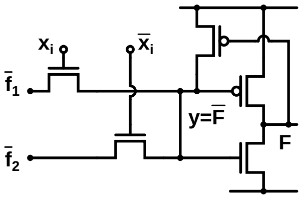


How

- The high output of the multiplexer (y) cannot rise above V_{dd} - V_{Tn}because we use nMOS multiplexers.
- Consequently, the pMOS transistor in the buffer inverter never quite turns off.
- This results in static power consumption in the inverter.

This can be avoided by adding a pull up pMOS with the inverter.

Use of Pull-up PMOS



- When the multiplexer output (y) is 'low', the inverter output (F) is high. The pMOS is off and has no effect.
- When the multiplexer output (y) goes 'high', the inverter output falls and turns the pMOS on.

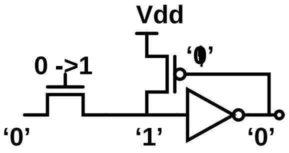
Now, even though the multiplexer nMOS turns 'off' as y approaches V_{dd} - V_{Tn}, the pMOS remains 'on' and takes the inverter input (y) all the way to V_{dd}.

This avoids leakage in the inverter.

Need for ratioing

The use of pMOS pull-up brings up another problem.

- Consider the equivalent circuit when the inverter output is ‘low’ and the pMOS is ‘on’.



If the final output is ‘low’, the pMOS pull-up is ‘on’. Now if the multiplexer output wants to go ‘low’, it has to fight the pMOS pull-up - which is trying to keep this node ‘high’.

In fact, the multiplexer n transistor and the pull up p transistor constitute a pseudo nMOS inverter.

Therefore, the multiplexer output cannot be pulled low unless the transistor geometries are appropriately ratioed.