








4

Basic Gates

Introduction

The basic logic gates available as primitives in the symbol palette of DSCH2 are listed below. We also give the corresponding symbol according to the IEEE standard, and the Verilog description.

Name	Logic symbol	Verilog description
AND		
NAND		
OR		
NOR		
XOR		
XNOR		
TRANSMISSION GATE		

The Nand Gate

The truth-table and logic symbol of the NAND gate with 2 inputs are shown below.

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

```
graph LR; in1[in1] --> nand2[nand2]; in2[in2] --> nand2; nand2 --> out1[out1];
```

Fig. 4-1. The truth table and symbol of the NAND gate

Logic simulation

- ❶ In DSCH2, select the NAND symbol in the palette, add two buttons and one lamp as shown above.
- ❷ Add interconnects if necessary to link the button and lamps to the cell pins.
- ❸ Verify the logic behavior of the cell.

CMOS Nand Gate Design

In CMOS design, the NAND gate consists of two nMOS in series connected to two pMOS in parallel. The schematic diagram of the NAND cell is reported below. The nMOS in series tie the output to the ground for one single combination $A=1, B=1$. For the three other combinations, the nMOS path is cut, but a least one pMOS ties the output to the supply VDD. Notice that both nMOS and pMOS devices are used in their best regime: the nMOS devices pass “0”, the pMOS pass “1”.

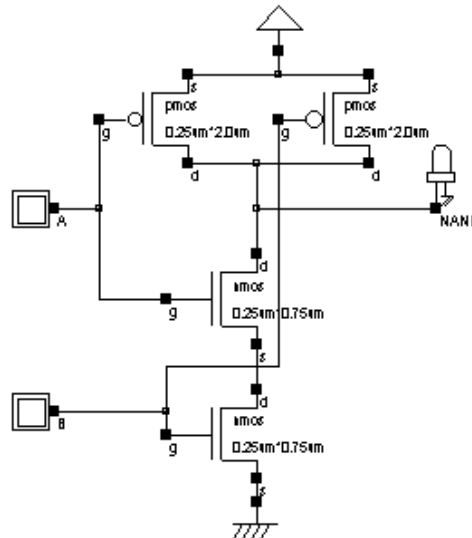


Fig. 4-2. The truth table and schematic diagram of the CMOS NAND gate design.

NAND Compiling

You may load the NAND gate design using the command **File -> Read->NAND.MSK**. You may also draw the NAND gate manually as for the inverter gate. An alternative solution is to compile directly the NAND gate into layout with Microwind2. In this case, complete the following procedure:

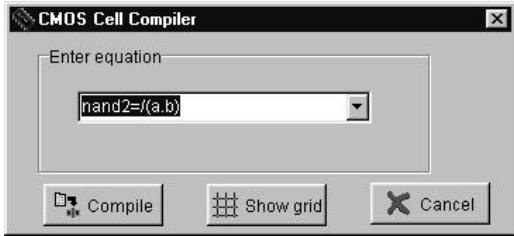
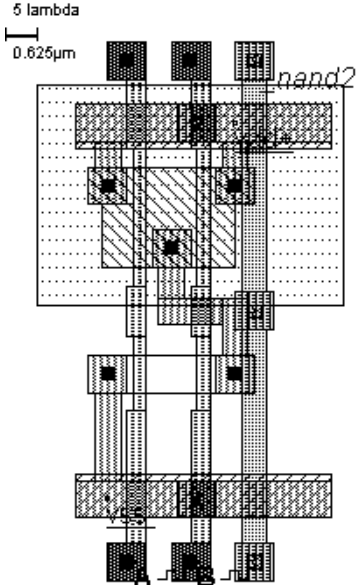
 <p>The CMOS Cell Compiler dialog box shows the equation <code>nand2=!(a.b)</code> entered in the 'Enter equation' field. Below the field are three buttons: 'Compile', 'Show grid', and 'Cancel'.</p>	<p>In Microwind2, click on Compile->Compile One Line. Select the line corresponding to the 2-input NAND description as shown above. The input and output names can be by the user modified.</p>
 <p>The layout shows a 2-input NAND gate. It features two input lines labeled 'A' and 'B' at the bottom, which connect to a network of transistors. The output line 'S' is at the top. The layout includes a grid and a scale bar indicating 5 lambda (0.625 μm). The text 'nand2' is placed near the top right of the layout.</p>	<p>Click "Compile". The result is reported above.</p> <p>The compiler has fixed the position of VDD power supply and the ground VSS. The texts A, B, and S have also been fixed to the layout. Default clocks are assigned to inputs A and B.</p>

Fig. 4-3. A NAND cell created by the CMOS compiler.

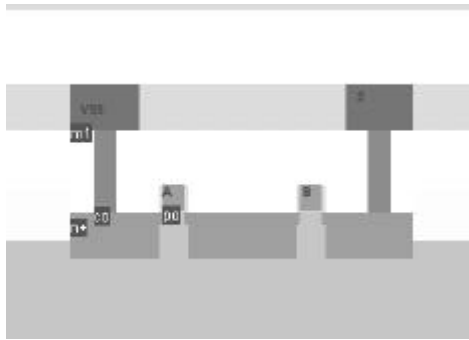
Fig. 4-4. Simulation of the NAND gate.

The rise time of the NAND2 gate is faster (15ps rather than 30ps) because of the pMOS devices in parallel.

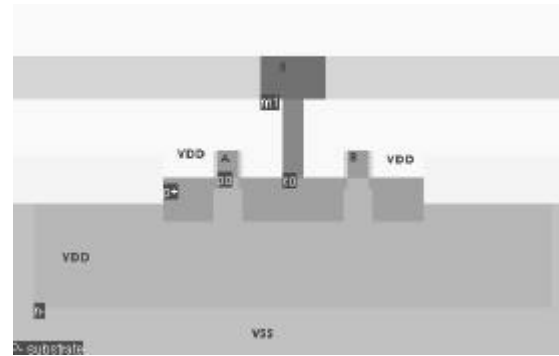
Inside the Nand gate



The 2D-process viewer is a useful tool to display the two nMOS in series and the two pMOS in parallel. Select the corresponding icon and draw an horizontal line in the layout in the middle of the nMOS channels. The figure below appears. In fig. 4-5a, the output is connected to the VSS supply only if A=1 and B=1. In fig. 4-5b, the output may be tied to VDD either when A=0 or B=0. Notice the n-well under the pMOS devices, polarized to VDD.



(a) nMOS devices



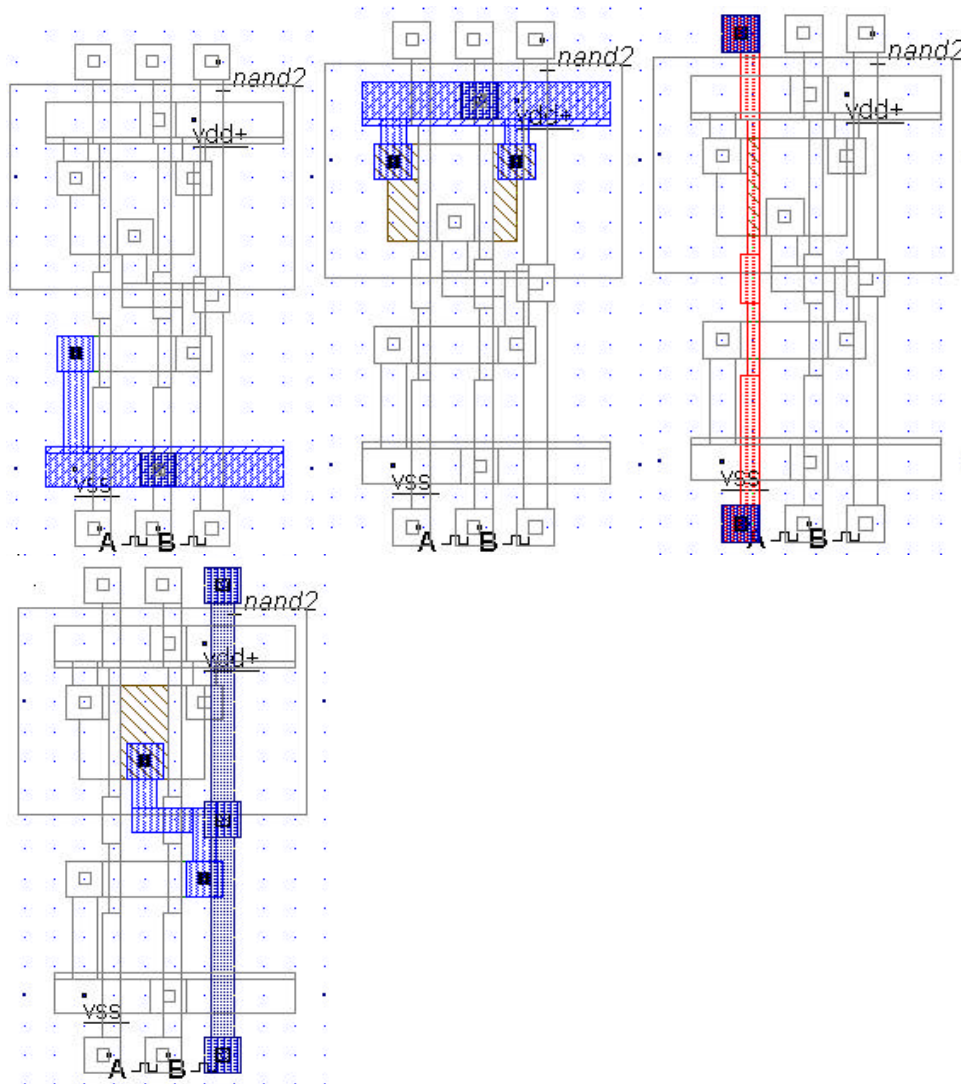
(b) pMOS devices

Fig. xxx. The nMOS devices in serial and the pMOS devices in parrallel

Electrical structure of the Nand Gate



The icon above is useful to get an insight of the electrical node structure of the layout. Select the icon and simply click inside the layout at the desired location. Information corresponding to the parasitic capacitance, resistance, as well as simulation properties are also displayed in a separate window. In figure xxx, the ground node, supply node, one input and one output are selected one after the other.



The ground supply

The VDD supply

The clock A

The output

Fig. xxx: Detail of the nodes

One single diffusion is better

The main advantage of joining the MOS devices diffusion whenever possible, rather than implementing all devices separately, is the silicon surface reduction, and consequently cost savings. A second advantage is the speed improvement. Joint diffusions lead to smaller areas, meaning lower parasitic capacitance, and thus shorter charge/discharge delays. The origin of the parasitic capacitance is mainly the N+/P-substrate junction capacitance due as the diode is polarized in inverse (P at low voltage VSS, N at higher voltage). Furthermore, the direct link between diffusions leaves space for metal routing.

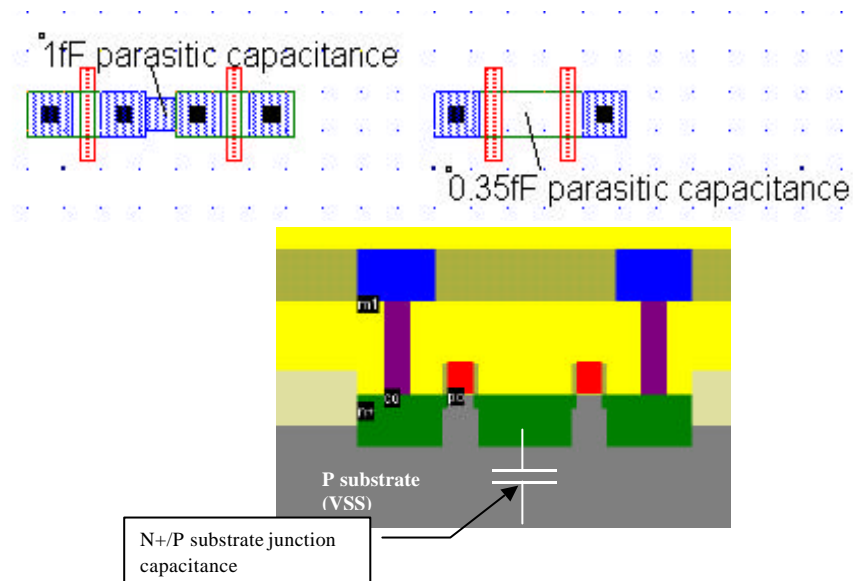


Fig. xxx: Joined diffusions lead to compact designs and speed improvements

Choosing the optimum pMOS placement

There are two solutions in implementing the pMOS devices, according to schematic diagram of figure xxx. One solution consists in placing the pMOS with two connections to VDD (solution 1), the second one with two connections to the output (solution 2). Both solutions are reported in the figure below. From the simulation of both structures, it can be seen that the structure with minimum diffusion and metal connected to the output, switches faster than the structure with larger output structure.

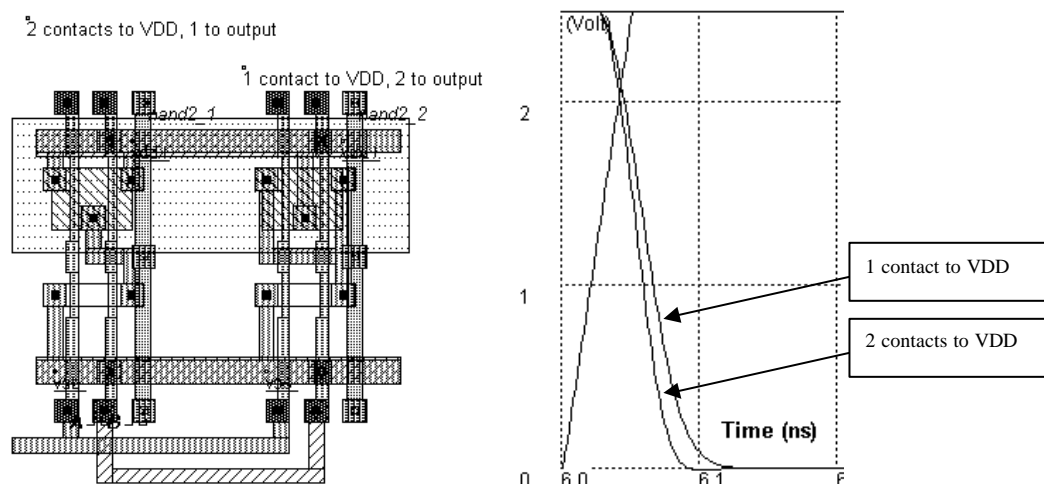


Fig. xxx: A minimum path for the output is preferred for an optimum speed

Cell Power Supply

The cell architecture has been optimized for easy supply and input/output routing. The supply bars have the property to connect naturally to the neighboring cells, so that specific effort for supply routing is not required. As an illustration, the figure below corresponds to the placement of three NAND gates. The VSS and VDD rails are common to all cells.

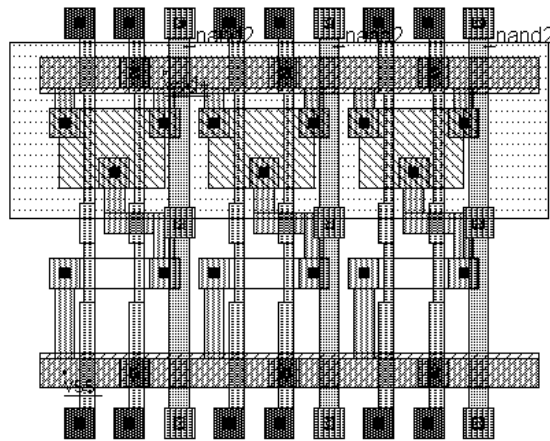
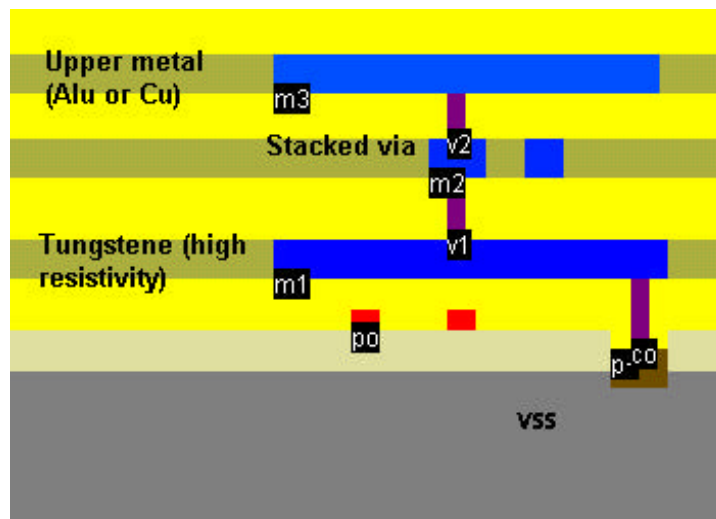


Fig. xxx : automatic abutment <xxx??> of VDD and VSS rails



Use the 2D-process viewer to display vertical structure of the supply rails. On the lower right hand side of the screen, the local substrate polarization to VSS appears. The VSS network consists in two parallel lines made from metal1 and metal3. The metal1 is highly resistive (Tungsten) so that it can only be used for short local routing. This is why a supplementary metal layer (level 3) is assigned to VSS, featuring low resistivity. Regularly, stacked via are placed to ensure a proper connection between the upper and lower metal layers. Notice that both polysilicon gates and input/output metal 2 interconnect may be routed easily inside the structure.



Cell Input/output

The input/output nodes are routed on the top and the bottom of the active parts, with a regular spacing to ease automatic channel routing between cells. Click **Compile -> Show grid** to superpose the routing grid on the layout. In figure xxx, the routing of input/output between basic cells is presented. Notice that the routing is performed either on the top or on the bottom of the active parts.

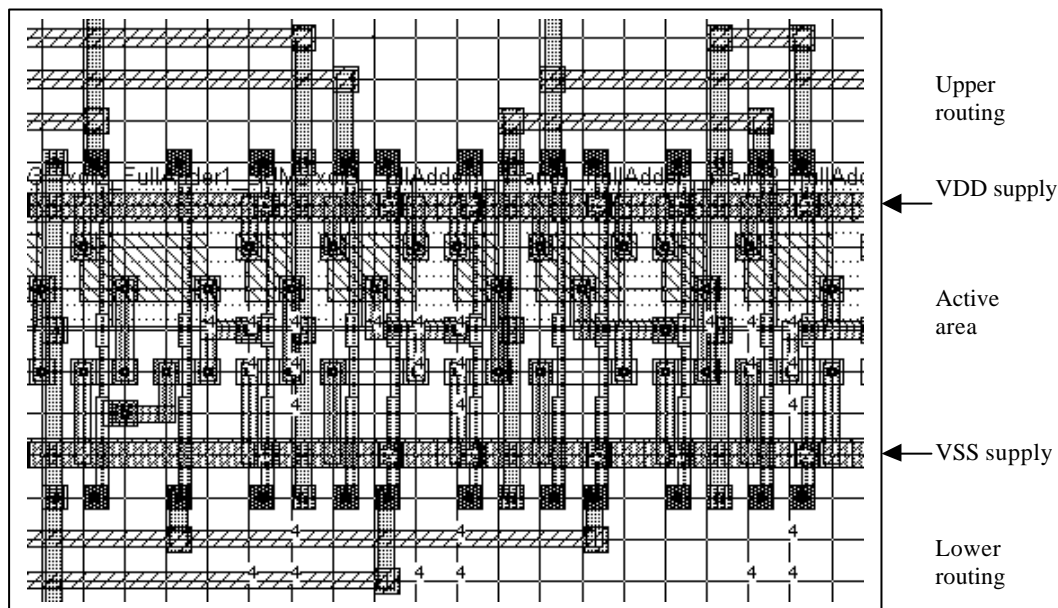


Fig. xxx : <Etienne update> Example of cell routing using a fixed grid

Sub-micron vs. Deep sub-micron technology

In $0.8\mu\text{m}$ technology, the design rules concerning the design of a link between polysilicon and metal2 layers are leading to the layout displayed on the left side of the figure below. Notice that the via plug between metal1 and metal2 is slightly larger than the contact from polysilicon to metal. In $0.25\mu\text{m}$ technology, the contacts have the same dimension and can be stacked one the top of each other. Consequently, the routing can be more dense and the cell design can be compacted.

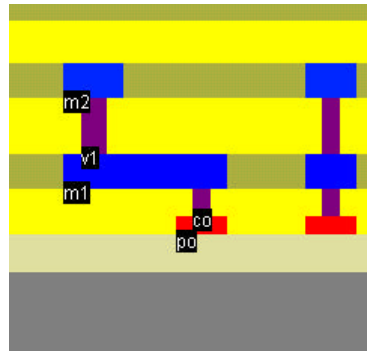


Fig. xxx : Conventional via (left) and stacked via (right)

The benefits of deep sub-micron technology are illustrated below. On the left side, the NAND gate compiled with the $0.8\mu\text{m}$ technology parameters (cmos08.rul) is drawn. On the right side, the same NAND gate compiled with the $0.25\mu\text{m}$ technology parameters (cmos025.rul) is reported. The gain in surface is impressive. The major reasons for this improvement are:

- a smaller routing pitch: 12 lambda in $0.8\mu\text{m}$, 10 lambda in $0.25\mu\text{m}$, thanks to more aggressive rules, specifically for the size of contacts
- the possibility to stack via: 2 routing pitch are required in $0.8\mu\text{m}$ to connect polysilicon to metal2, one single pitch is required in $0.25\mu\text{m}$

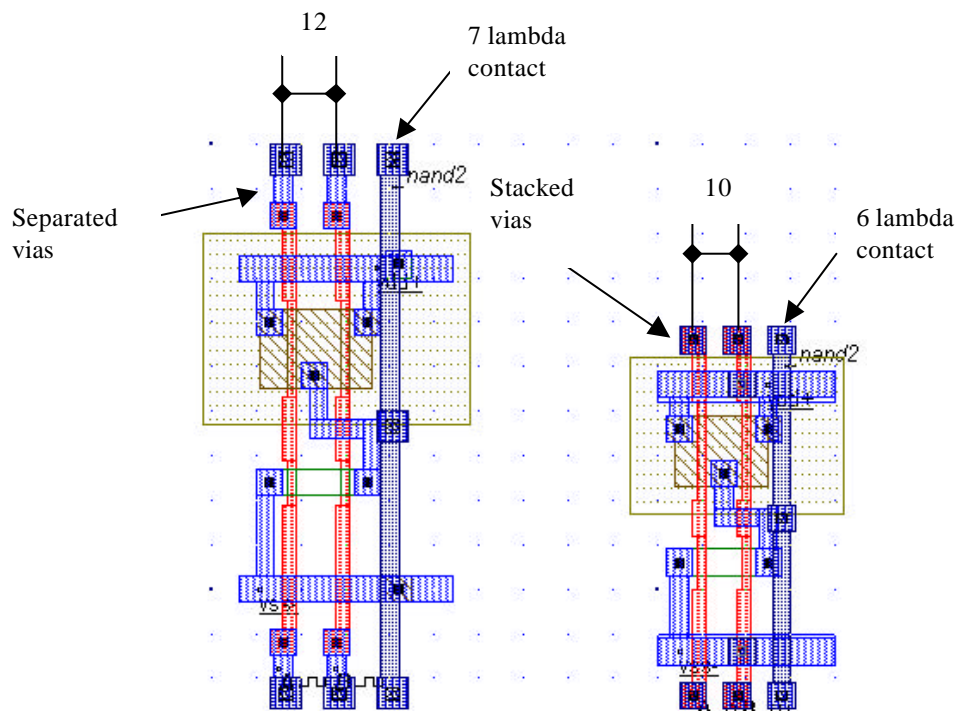


Fig. xxx: silicon surface improvement in deep-submicron technology

The AND gate

As can be seen in the schematic diagram and in the compiled results, the AND gate is the sum of a NAND2 gate and an inverter. The layout ready to simulate can be found in the file “AND2.MSK”. In CMOS, the negative gates (NAND, NOR, INV) are faster and simpler than the non-negative gates (AND, OR, Buffer). The cell delay observed in the figure xxx are significantly higher than for the NAND2 gate alone, due to the inverter stage delay.

Notice that a more compact layout of the AND cell may be found by joining the diffusions of the NAND and the Inverter cells. This would save one horizontal routing pitch and improve the cell speed.

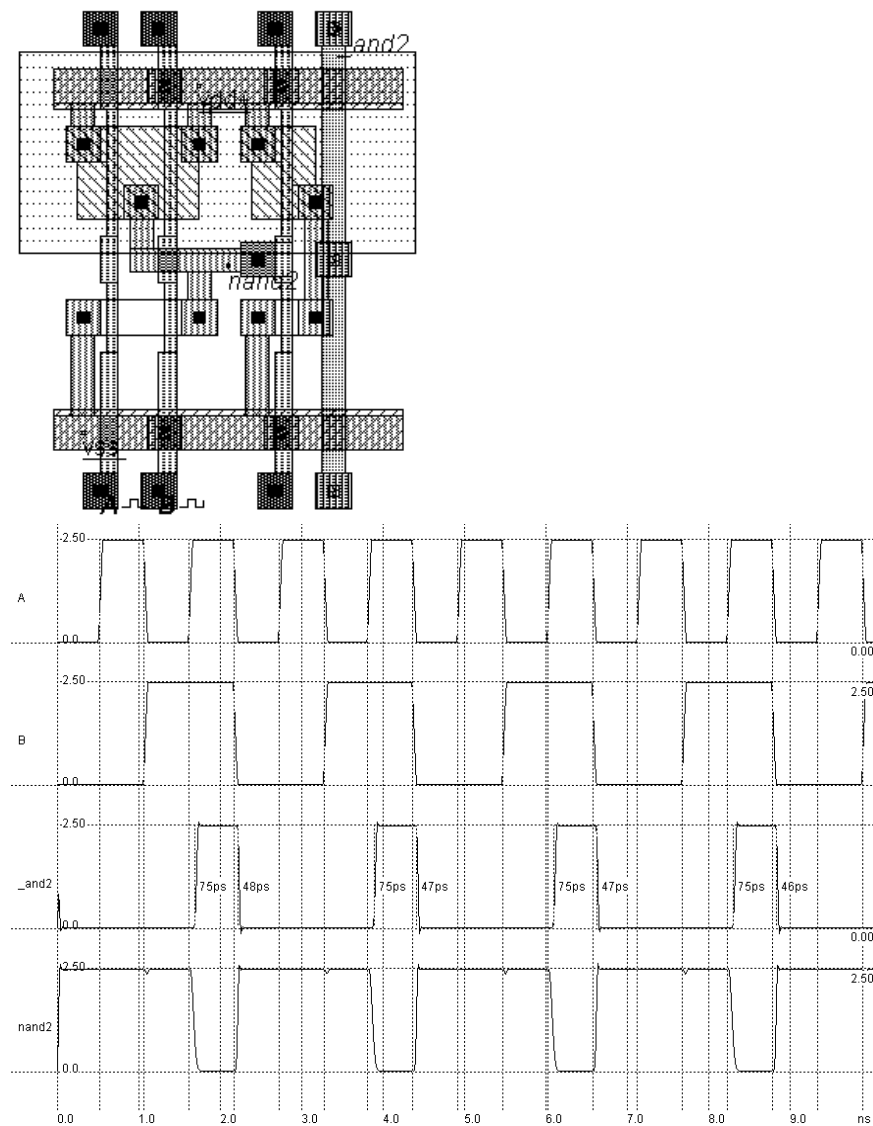


Fig. xxx: Layout and simulation of the AND gate

The 3-Input OR Gate

The truth-table and the schematic diagram of the three-input OR gate are shown in Figure xxx. You may use the DSCH2 logic editor to design a schematic diagram based on the OR gate, generate a Verilog description, and compile the text file in Microwind2. As can be seen again in the final layout, the OR gate is the sum of a NOR3 gate and an inverter.

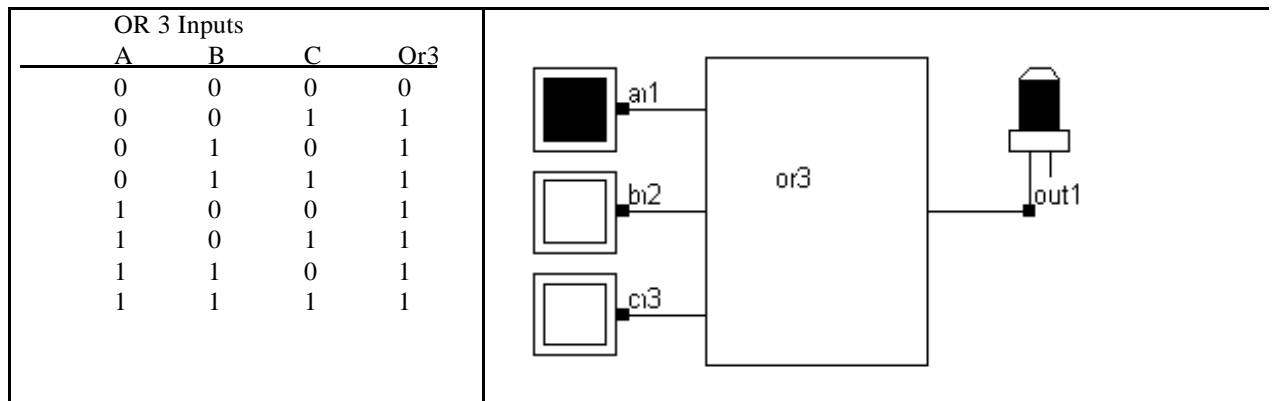


Fig. xxx. The truth table and symbol of the OR3 gate

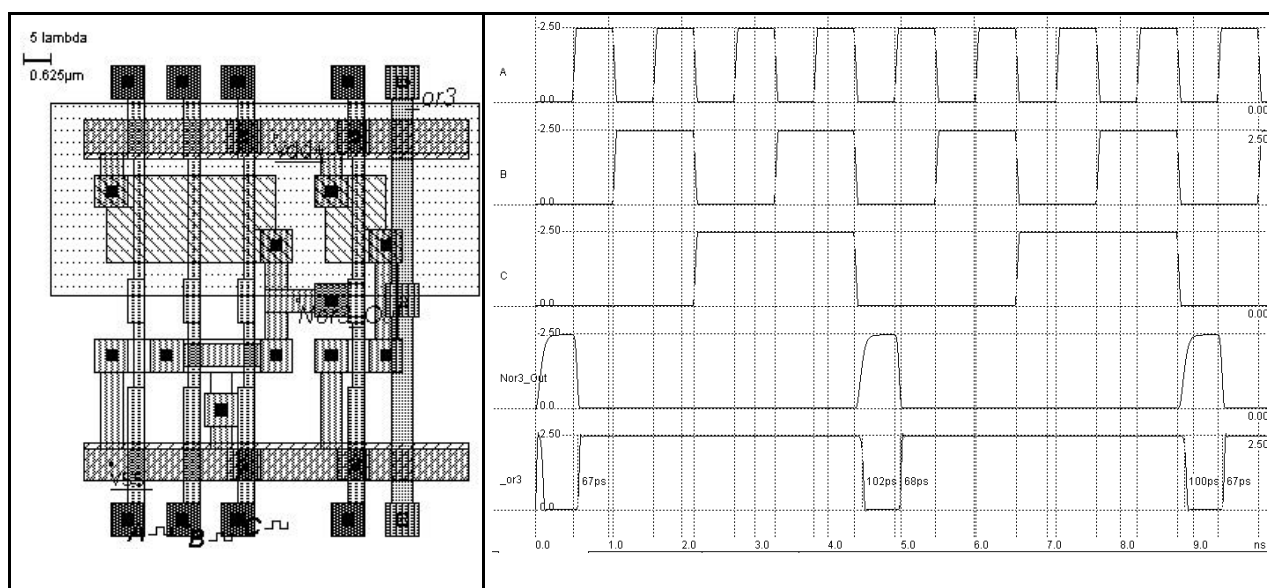


Fig. xxx. Layout and simulation of the OR3 gate (OR3.MSK).

About the n-Input NOR Gate

The 5-input or 6-input NOR gate designs may be deduced from the 3-input NOR gate design, but a problem rises. The pMOS devices in series lead to very poor output node charge to VDD, and consequently poor rise time performances, as shown below. Meanwhile, the nMOS device in parallel

provoke very short discharge to ground, meaning fast fall time. This non-symmetrical behavior can be tolerate up to a certain limit. This is why the n-input NOR gates ($n > 4$) are built from separated stages.

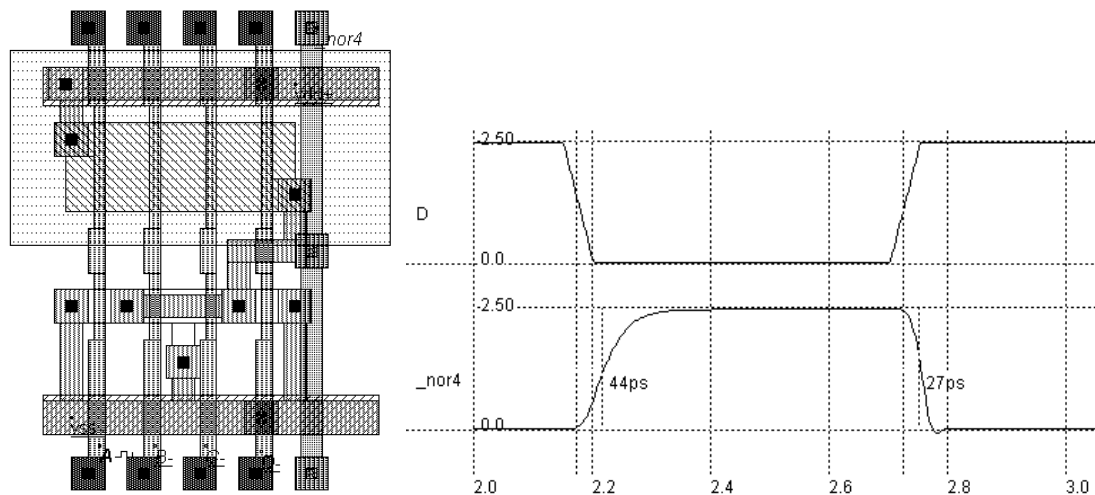



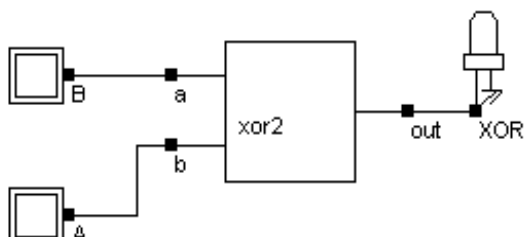
Fig. xxx : the non-symetrical behavior of the NOR4 gate

The XOR Gate



XOR 2 inputs

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0



The truth-table and the schematic diagram of the CMOS XOR gate are shown in Figure xxx. There exist many possibilities for implementing the XOR function into CMOS.

A poor design

The least efficient design, but the most forward, consists in building the XOR logic circuit from its Boolean equation:

$$\text{XOR} = A.B + A.\bar{B}$$

The direct translation into primitives is reported below. It requires two inverters, two AND gates and one OR gate.

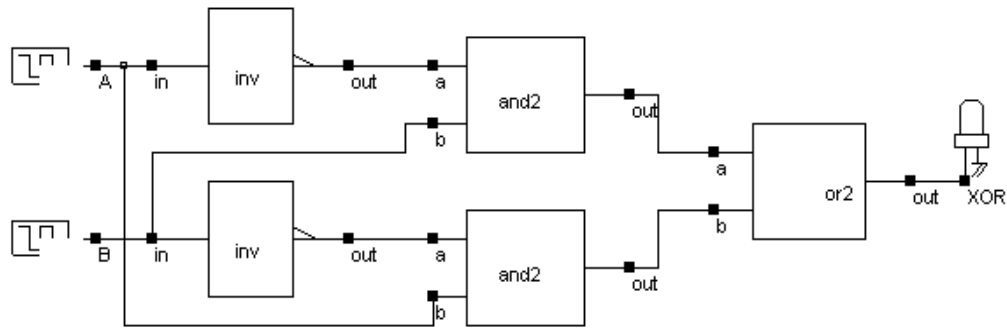


Fig. xxx. The direct translation of the XOR function into primitives

When compiled into layout, the function represents 22 transistors. The layout is quite complex when compiled by Microwind2. From left to right, we recognize two inverters (B first, A second), two AND gates, and finally a 2-input NOR gate, producing the result at the right hand side of the block. The simulation shows typical delays of 170-220 ps in 0.25 μ m technology.

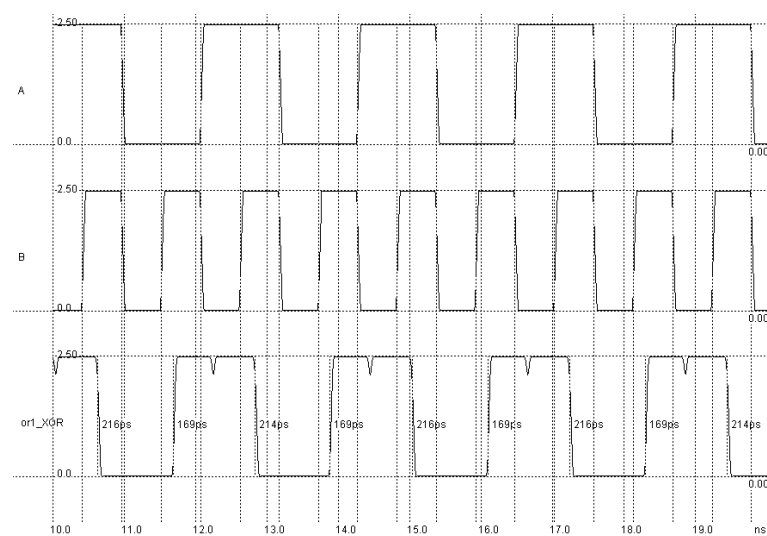
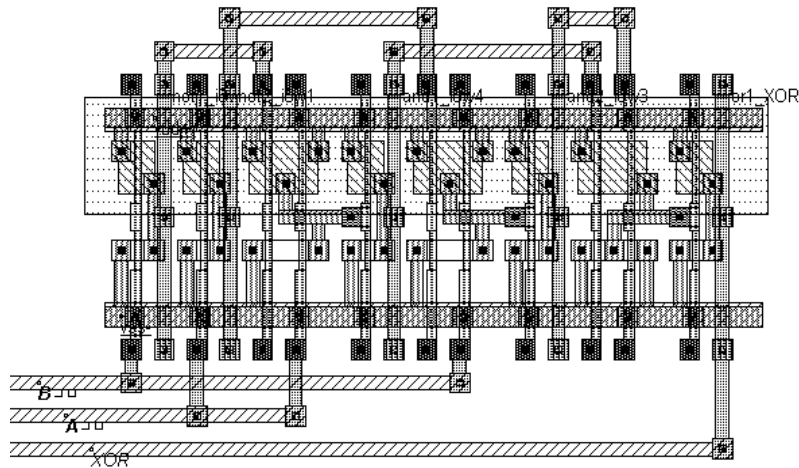


Fig. xxx. Compilation and simulation the XOR gate

An efficient design

The proposed solution consists of a transmission-gate implementation of the XOR operator. The truth table of the XOR can be read as follow: IF B=0, OUT=A, IF B=1, OUT = Inv(A). The principle of the circuit presented below is to enable the A signal to flow to node N1 if B=1 and to enable the **Inv(A)** signal to flow to node N1 if B=0. The node **OUT** inverts N1, so that we can find the XOR operator. Notice that the nMOS and pMOS devices situated in the middle of the gate serve as pass transistors.

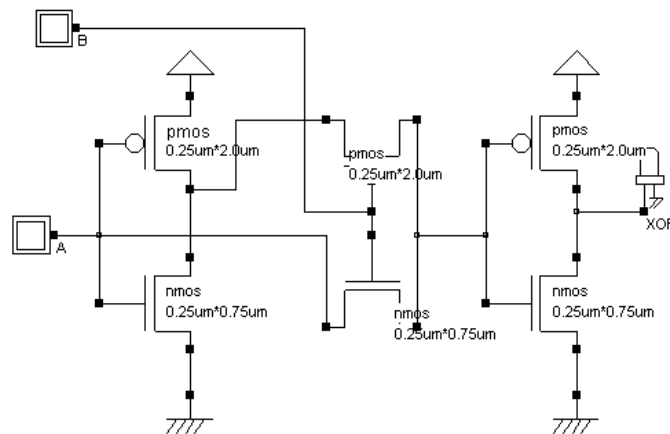


Fig. xxx. The schematic diagram of the XOR gate (XORCmos.SCH)

You may use DSCH2 to create the cell, generate the Verilog description and compile the resulting text. In Microwind2, the Verilog compiler is able to construct the XOR cell as reported in Figure xxx. You may add a visible property to the intermediate node which serves as an input of the second inverter. See how the signal, called “internal”, is altered by V_{tn} (when the nMOS is ON) and V_{tp} (when the pMOS is ON). Fortunately, the inverter regenerates the signal.

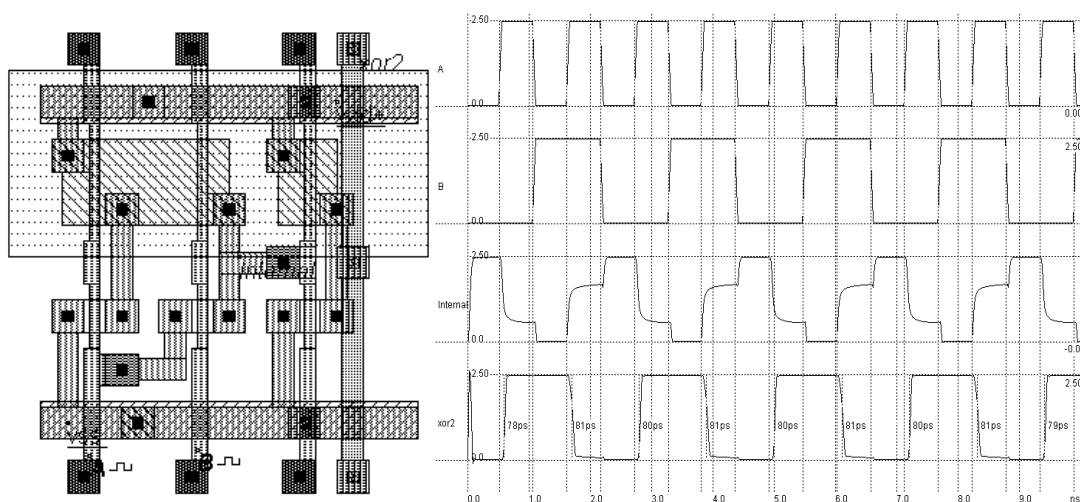


Fig. xxx. Layout and simulation of the XOR gate (XOR.MSK).

Complex Gates

The complex gate design technique applies for any combination of operators **AND** and **OR**. The technique produces compact cells with higher performances in terms of spacing and speed than conventional logic circuits. To illustrate the concept of complex gates, let us take the example of the following Boolean equation:

$$F = \neg(A + (B.C))$$

The logic circuit corresponding to this equation is reported below. The circuit is built using a 2-input NOR and a 2-input AND cell, that is 10 transistors and three delay stages.

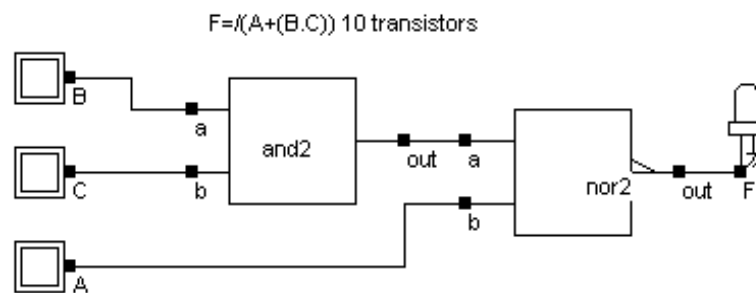


Fig. xxx: The conventional schematic diagram of the function $F = \neg(A + (B.C))$

A much more compact exists in this case (Figure xxx), consisting in the following steps:

1. For the nMOS network, translate the AND operator '.' into nMOS in series, and the OR operator '+' in nMOS in parallel.
2. For the pMOS network, translate the AND operator '.' into pMOS in parallel, and the OR operator '+' in pMOS in series.
3. If the function is non-inverting, as for ' $F = A + (B.C)$ ', an inverter is mandatory.

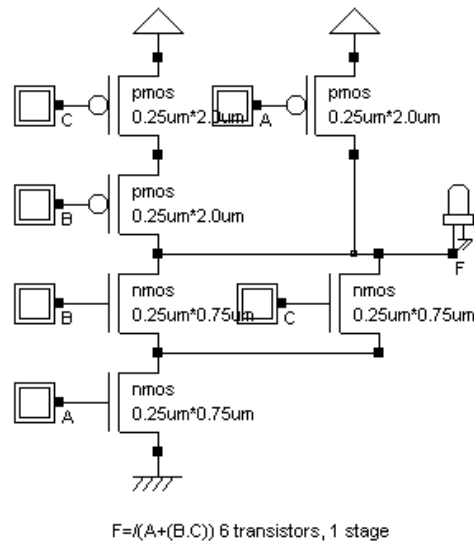
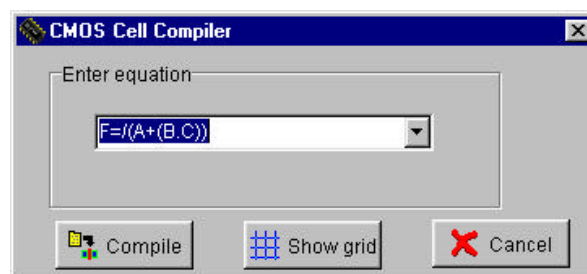


Fig. xxx: The complex gate implementation of the function $F = /(A+(B.C))$

Microwind2 is able to generate the CMOS layout corresponding to any description based on the operators **AND** and **OR**, using the command **Compile -> Compile one line**. Using the keyboard, enter the cell equation, or modify the items proposed in the list of examples. In the one-line equation, the first parameter is the output name. In the present case that name is **s**. The sign '=' is obligatory. The '/' sign corresponds to the operation NOT and can be *used only* right after the '=' sign. The parenthesis '(' ')' are used to build the function, where '.' is the AND operator and '+' is the OR operator.



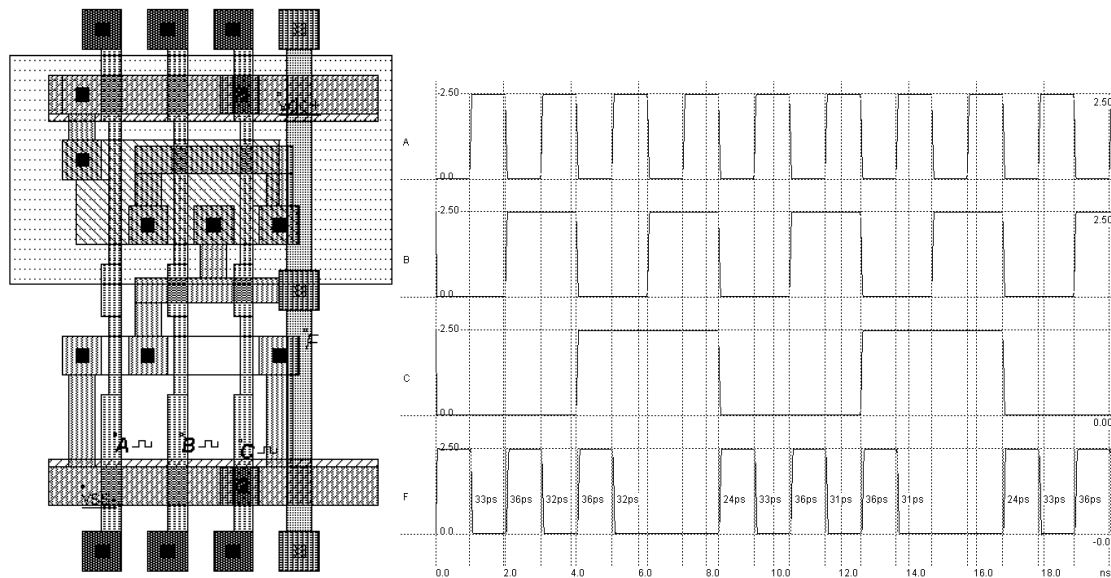


Fig.xxx. A compiled complex gate and its analog simulation (ComplexABC.MSK)

CELL	EQUATION EXAMPLE	CELL	EQUATION EXAMPLE
<i>Inverter</i>	$\text{out} = \text{in}$	<i>Buffer</i>	$\text{Buff} = \text{in}$
<i>NAND gate</i>	$n = \text{/(a.b)}$	<i>AND gate</i>	$s = \text{a.b}$
<i>OR gate</i>	$s = \text{a+b+c}$	<i>AND-OR Gate</i>	$\text{Andor} = \text{A.(B+C)}$
<i>CARRY Cell</i>	$\text{cout} = (\text{a.b}) + (\text{cin.}(\text{a+b}))$		

Multiplexor

Multiplexing means transmitting a large amount of information through a smaller number of connections. A digital multiplexer is a circuit that selects binary information from one of many input logic signals and directs it to a single input line. The main component of the multiplexer is a basic cell called the transmission gate. The transmission gate let a signal flow if Enable is asserted. Remember that the n-channel MOS is only good for low signals, and the p-channel MOS is only good for high signals. To pass logic signals well, both a n-channel device and a p-channel device must be used, as shown in figure xxx. The main drawback is the need for two control signals Enable and /Enable, thus an inverter is required.

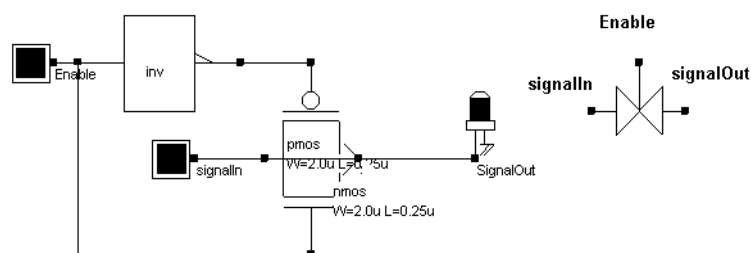


Fig.xxx. The transmission gate used as a multiplexor

In DSCH2, a transmission gate symbol exists, as shown in the right part of figure xxx. It includes the nMOS, pMOS and inverter cells. Concerning the layout, the channel length is usually the minimum length available in the technology, and the width is set large, in order to reduce the parasitic 'on' resistance of the gate.

4 to 1 Multiplexer

The multiplexer is a very useful function and has a multitude of application. The selection of a particular input line is controlled by a set of selection lines. Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected. Figure xxx shows the transmission gate implementation of the 4 to 1 multiplexer. In the configuration $S1=1$, $S2=0$, the input 'C' is connected to the output.

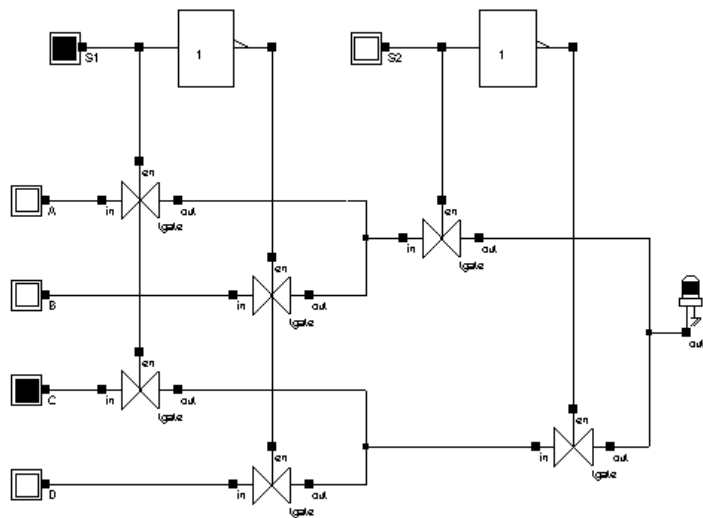


Fig. xxx 4 to 1 multiplexing based on transmission gates (Mux4to1.sch)

Figure xxx shows the n-channel MOS implementation of the 4 to 1 multiplexer. The result is simpler than for the transmission gate implementation, works faster, requires fewer devices. The major drawback is the waveform degradation of high signals due to the threshold voltage drop. Therefore, the output should later be refreshed thanks to a buffer.

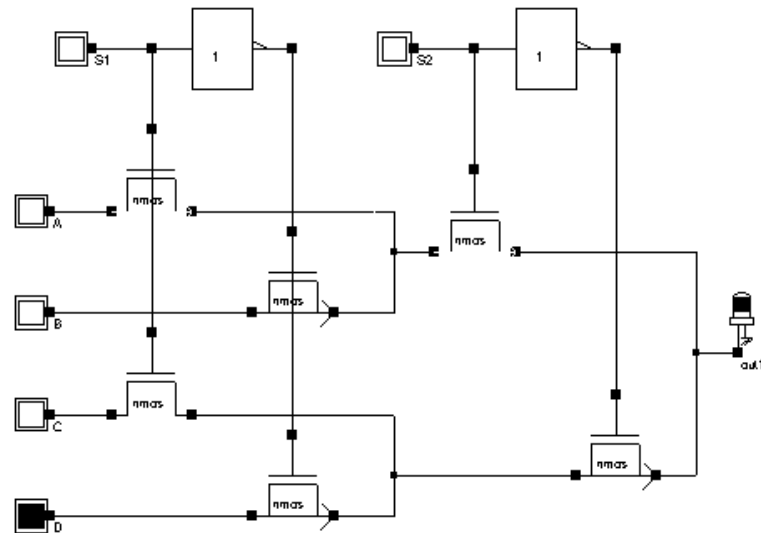


Fig. xxx 4 to 1 multiplexing based on MOS devices (Mux4to1Mos.sch)

Keyboard Multiplexer

Figure xxx gives an example of 2 multiplexed hexadecimal keyboards sharing the same hexadecimal display, using transmission gates. We use a clock to generate an alternative selection of the keyboard information.

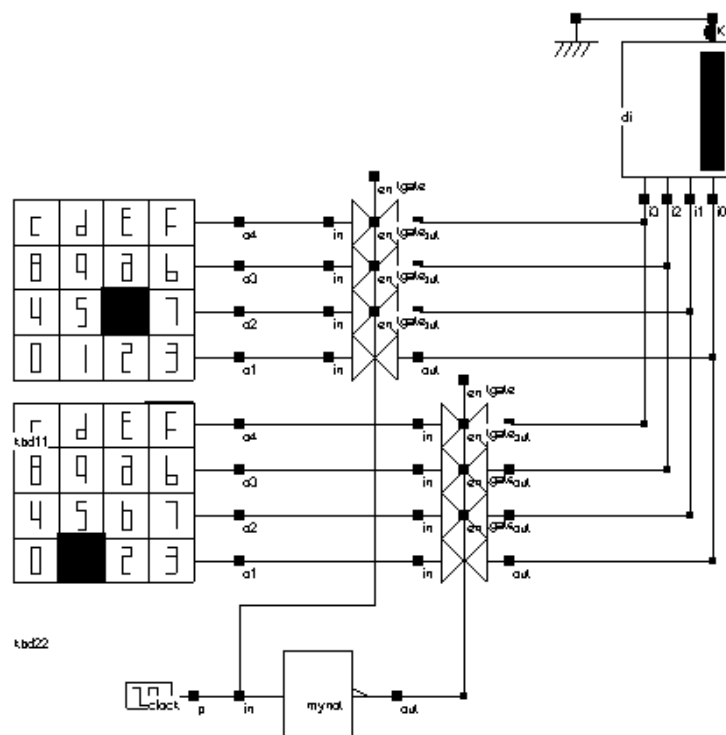


Fig. xxx Keyboard multiplexing based on transmission gates (Mux2Kbd.sch)

Conclusion

In this chapter, the design of basic cells has been reviewed. The specific design of the XOR gate has been detailed, as well as the complex gates.