CS220 Computer Architecture
Digital Logic Design
Practical 6

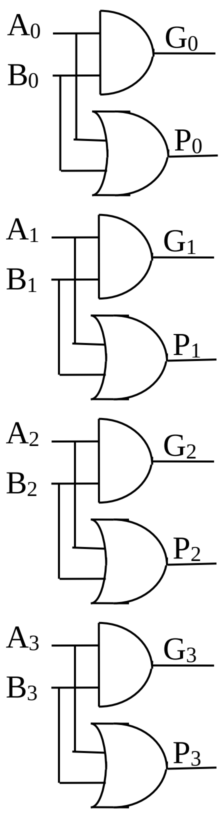
The TkGate Logic Simulator under Linux is used to implement these practicals. **Boot Linux, log in and launch the TkGate application.**

4-bit Carry Look-Ahead Adder

The objective of this circuit is to generate carry-outs for each bit position simultaneously so that the sum for each bit position can be computed immediately as opposed to other adder designs where carry bits need to ripple through from lower bit positions leading to longer propagation delays for the circuit and lower performance.

The design of the look-ahead carry generator involves two Boolean functions named Generate and Propagate. For each of the pair wise bits of the number these functions are defined as:

$G_i = A_i . B_i$; Carry will definitely be generated if A_i=1 and B_i=1
$P_i = A_i + B_i$; Carry might be generated when this function is combined with c-out_{i-1}



The **first stage** in developing a 4-bit look-ahead adder

The carry-out bit c-out$_i$ is generated whenever there are two or more 1s present when adding two bits A$_i$ and B$_i$ of a number combined with the value of a carry in from the lower bit position. In terms of G$_i$ and P$_i$, a carry out is generated whenever:-

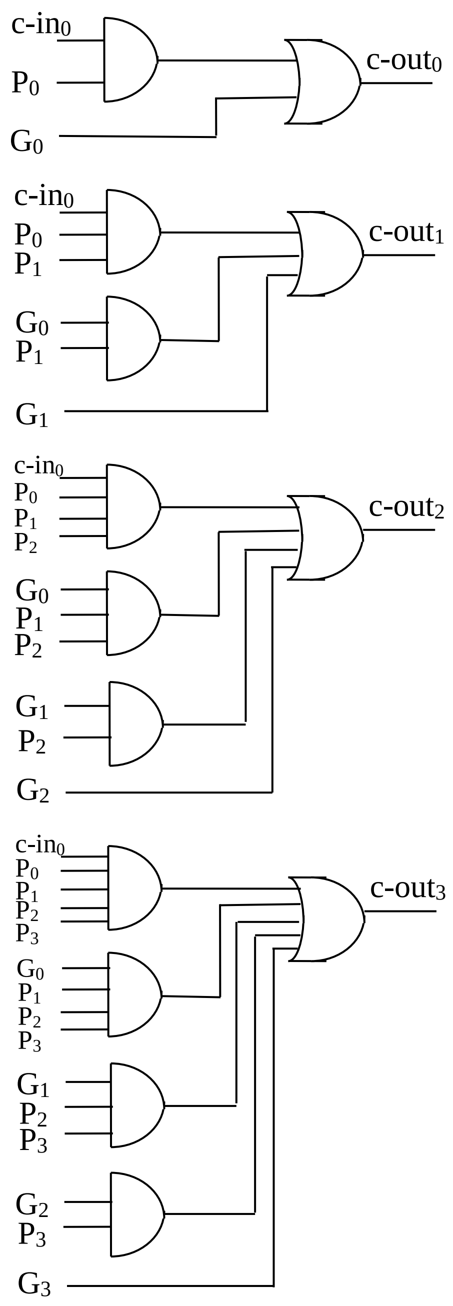
1) The corresponding function G$_i$ is 1 **or**
2) The c-out$_{i-1}$=1 and the function P$_i$ = 1 simultaneously.

Therefore, the carry-out bit corresponding to a pair of bits A$_i$ and B$_i$ is calculated according to the equation:

$$\text{c-out}_i = G_i + P_i.\text{c-in}_i$$

For a four-bit adder the carry-outs are calculated as follows

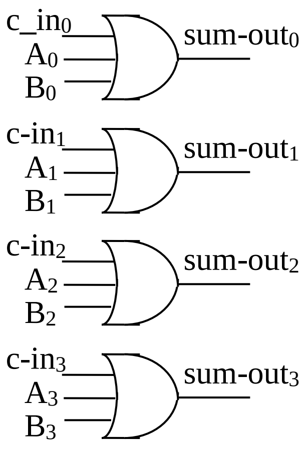
$$\text{c-out}_0 = G_0 + P_0.\text{c-in}_0$$
$$\text{c-out}_1 = G_1 + P_1(G_0 + P_0.\text{c-in}_0) = G_1 + P_1G_0 + P_1P_0.\text{c-in}_0$$
$$\text{c-out}_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0.\text{c-in}_0$$
$$\text{c-out}_3 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0.\text{c-in}_0$$



The **second stage** in developing a 4-bit look-ahead adder – parallel generation of carry outs for all bit positions

Finally, the sums can be calculated from the following equations, where c-in is taken from the carry calculated in the circuit in stage 2.

$\text{sum\_out}_0 = A_0 \oplus B_0 \oplus \text{c\_in}_0$
$\text{sum\_out}_1 = A_1 \oplus B_1 \oplus \text{c\_in}_1$
$\text{sum\_out}_2 = A_2 \oplus B_2 \oplus \text{c\_in}_2$
$\text{sum\_out}_3 = A_3 \oplus B_3 \oplus \text{c\_in}_3$



Stage 3 of the carry look-ahead adder – Generation of the sums

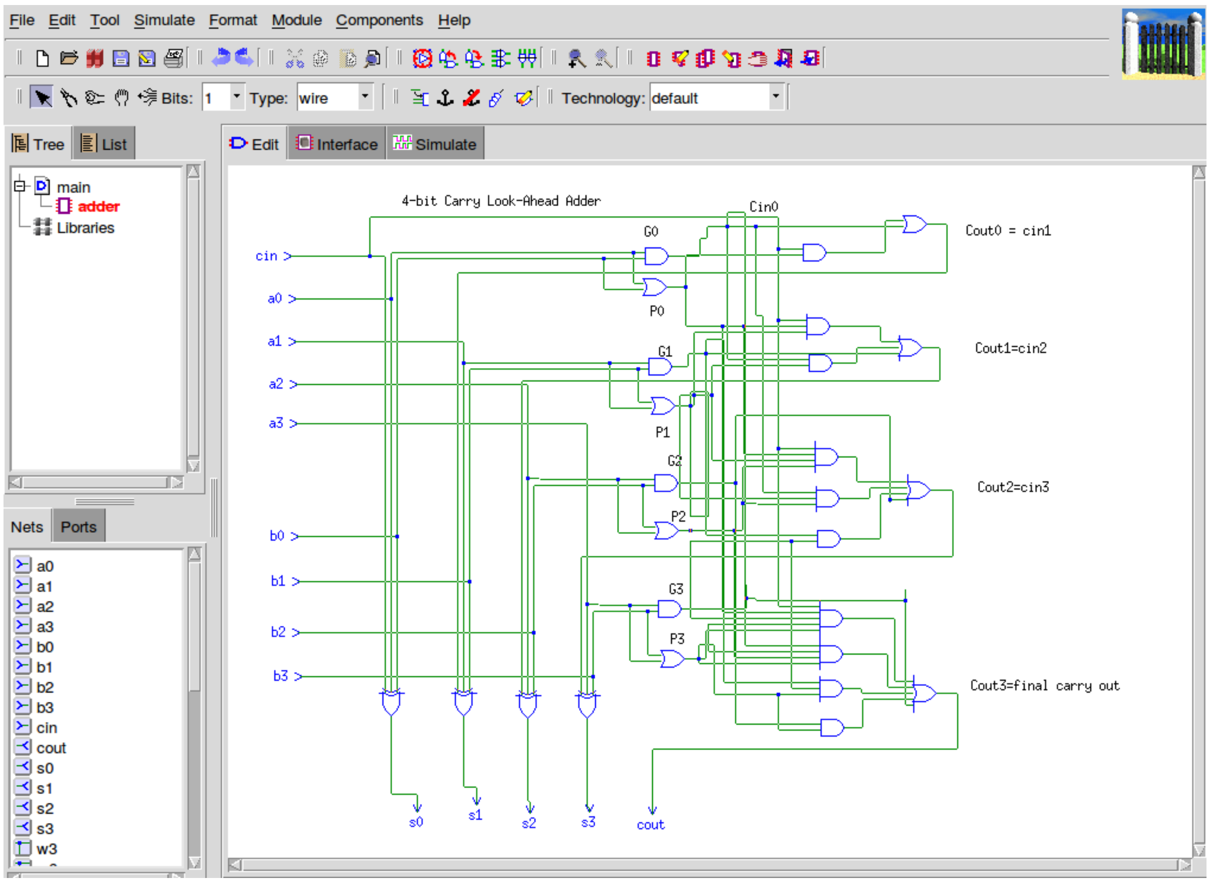
By putting all the stages of the circuit together, you can see that the critical path through the circuit involves four gate delays before the complete sum of the four-bit numbers A+B is calculated. This represents a significant improvement in circuit performance over the ripple carry adder design.

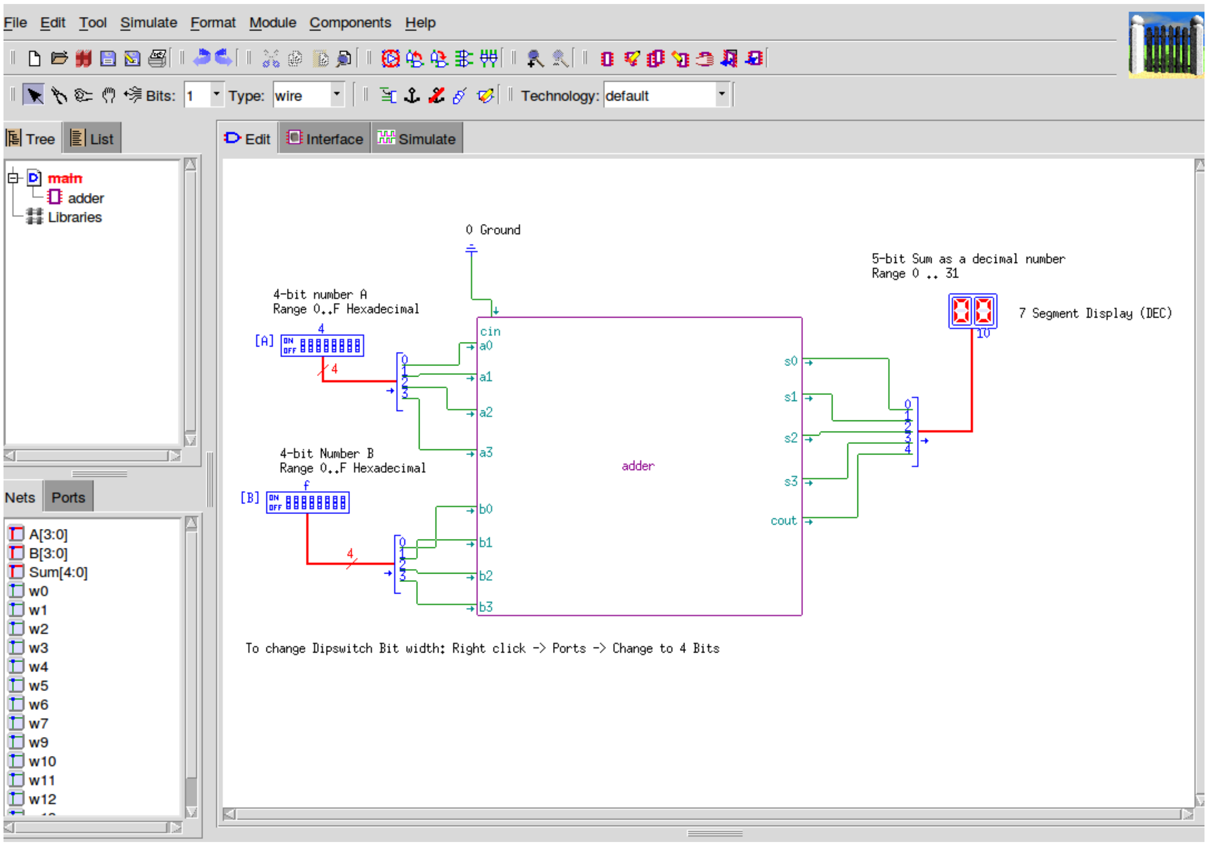
TO DO
Create the 4-bit look-ahead adder circuit **AS A MODULE** with 9 inputs (initial carry-in and 2x4bit numbers) and 5 outputs (4-bit sum and 1-bit final carry-out).

When the module implementation is complete, create an instance of it in the main edit window.

The 4-bit numbers to be added can be supplied by dip-switches and the sum can be displayed on a decimal display. Test your circuit to verify that it adds two 4-bit numbers correctly.

To add larger numbers, the adder module could be daisy chained by connecting its carry out to the carry in of the next module.





To change Dipswitch Bit width: Right click -> Ports -> Change to 4 Bits