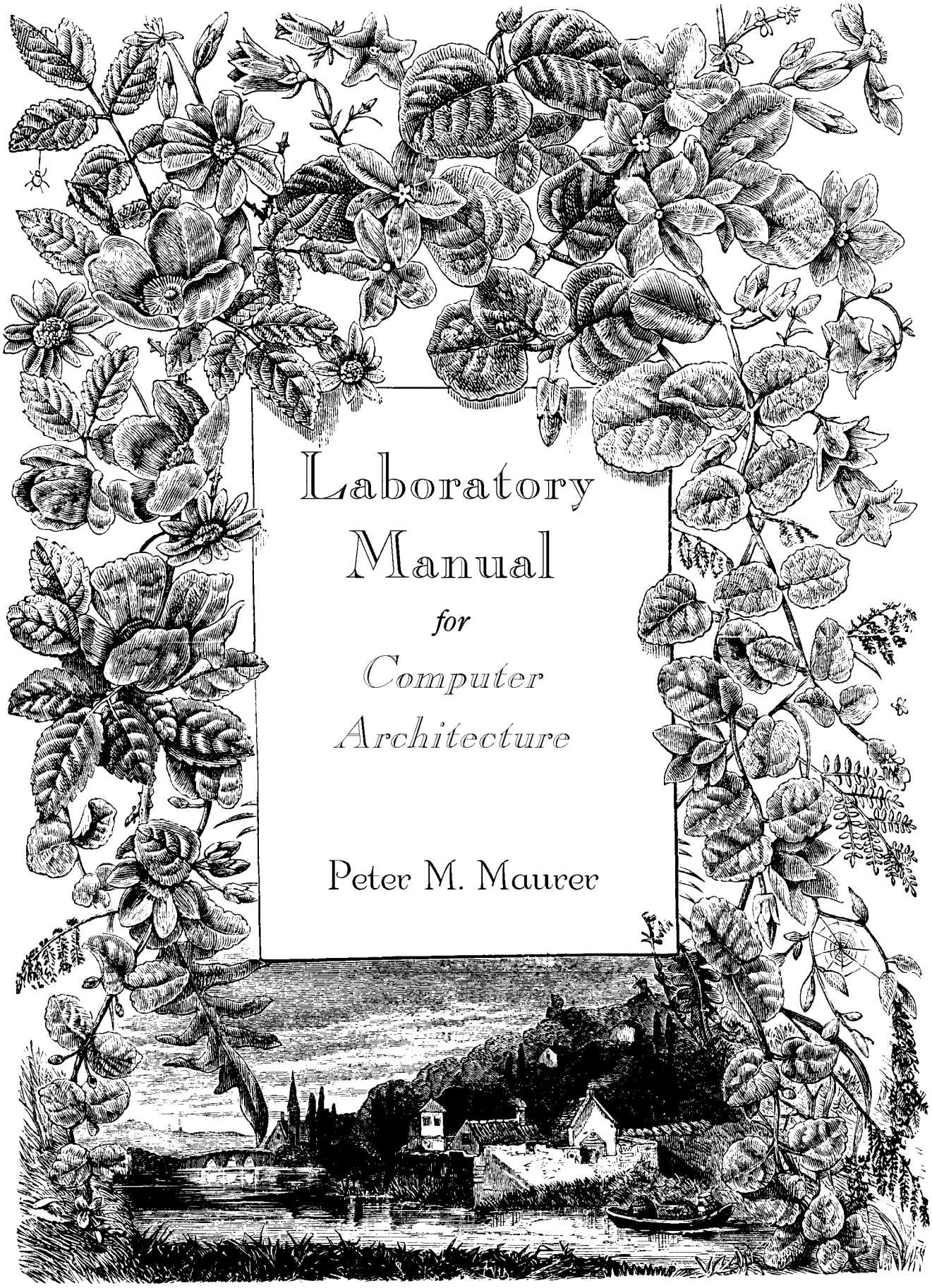


Laboratory Manual *for* *Computer Architecture*

Peter M. Maurer

Prolog
The FHDL System

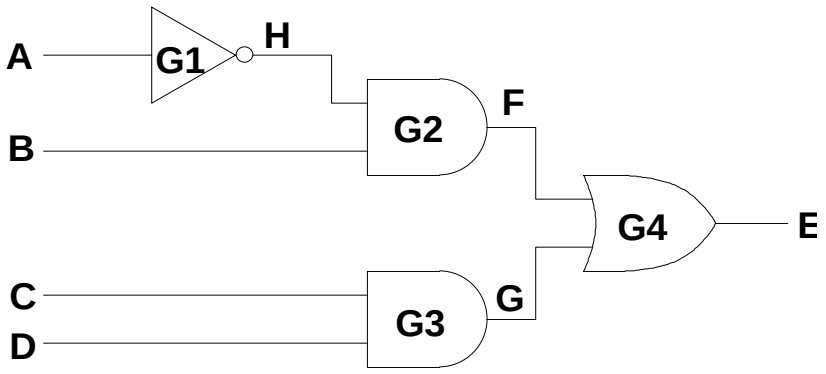
Each computer in your laboratory should have **WinFHDL** installed on it. **WinFHDL** runs under Windows 3.1 or Windows 95, and usually resides in the **FHDL System** program group. The program icon will normally have the longer name **FHDL Logic Simulator**. To create a circuit, start this program and select the **New** command from the **File** menu. Type your circuit description into the window that appears.

Syntax: Each Florida Hardware Design Language (FHDL) statement has a label field, an operation code, and a list of operands. The following is a typical statement.

G1: and (a,b),c

The label field, which is optional, starts at the first character of a statement and ends with a colon (:). The operation code must always be present and must be preceded by one or more spaces or tabs. If a label is present, the operation code follows the label. The operand list, which is also optional, follows the operation code, and must be separated from the operation code by one or more spaces or tabs. Every statement must be followed by a newline character (return key) or a semicolon (;). If it is necessary to extend an FHDL statement across several lines, every line except the last must end in a comma.

User-defined labels and names may contain upper and lower case letters, digits, underlines, and dollar signs. To translate a logic diagram such as the following into FHDL, first choose unique names for each of the connections.



In this example the names **A-H** have been chosen for the connections. Next you must choose a unique name for the circuit (for this example we will choose example1.) The circuit description must begin with the following statement.

example1: circuit

Next make a list of the primary inputs and outputs of the circuit. In the example, **A, B, C**, and **D** are primary inputs while **E** is a primary output. An **input** statement is used to declare primary inputs, while an **output** statement is used to declare primary outputs. The following statements would be used for our example. (The keywords **input** and **inputs** are interchangable, as are **output** and
outputs.)

```
inputs   A, B, C, D
outputs  E
```

Next describe each gate using an appropriate statement. In the example, one not statement, two and statements, and one or statement will be required. These statements do not require labels, but it is a good idea to use them anyway. The gates of our example would be written as follows.

```
g1:  not    A,H
g2:  and    (H,B),F
g3:  and    (C,D),G
g4:  or     (F,G),E
```

The order of the statements (including the input and output statements) is arbitrary. Its possible to use more than one input or output statement. When this is done, the primary input and output lists are created by combining all names in the order that they are specified in the circuit. The input and output statements may appear anywhere in the circuit. The operand list of a gate has two parts. The first part lists the gate's inputs while the second part lists the gate's outputs. If there is more than one input, the list of inputs must be enclosed in parentheses. Similarly if there is more than one output, the list of outputs must be enclosed in parentheses. Finally, the circuit description must end with an **endcircuit** statement. The entire FHDL description is as follows.

```
example1:   circuit
      inputs    A,B,C,D
      outputs   E
g1:   not       A,H
g2:   and       (H,B),F
g3:   and       (C,D),G
g4:   or        (F,G),E
      endcircuit
```

FHDL provides many different kinds of gates. In most cases the order of the inputs and outputs is significant, although in this example, the order of the inputs does not matter. Subcircuits can be defined to provide gate types that are not provided by the FHDL system.

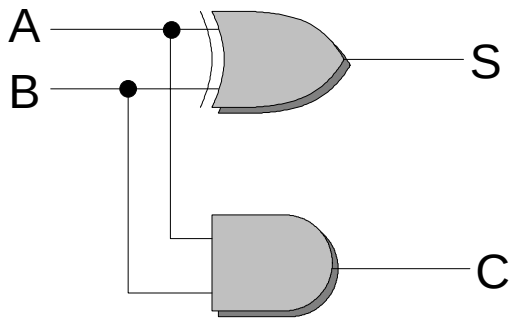
See the online help for more information.

Experiment 1
Ripple Carry Addition

Objective: Learn the principles of designing hierarchical circuits, and the principles of binary addition..

Project: In this experiment, you will create a 16-bit ripple carry adder/subtractor.

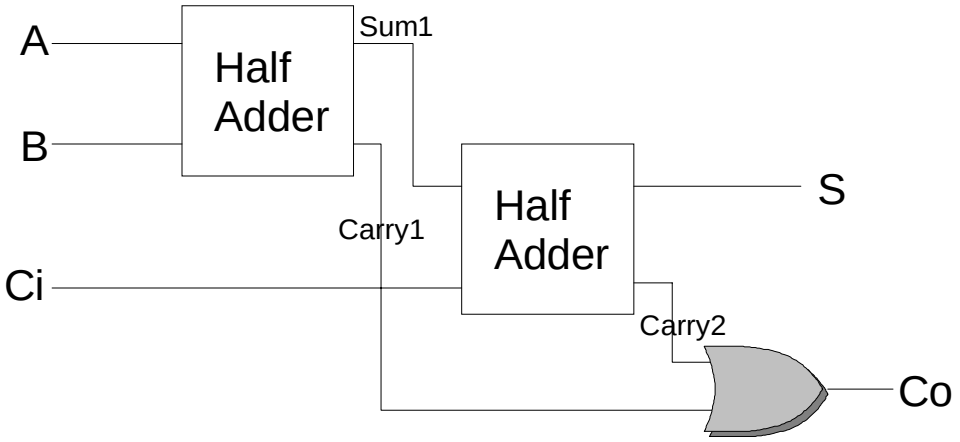
Procedure: Begin by designing a half-adder which implements the following logic diagram.



The FHDL equivalent of this circuit is the following.

```
HalfAdder: circuit
           inputs   A,B
           outputs  S,C
G1:        xor      (A,B),S
G2:        and      (A,B),C
           endcircuit
```

Enter and test this circuit before proceeding. The half adder can be used to construct a full adder as illustrated in the following diagram.

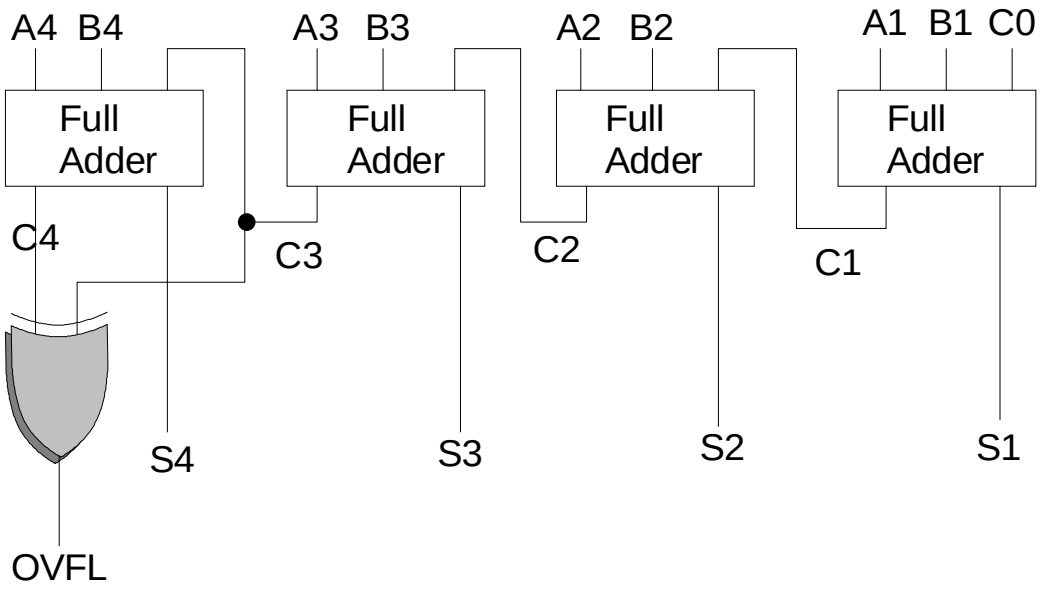


This circuit is translated into FHDL in the following manner.

```
FullAdder:  circuit
            inputs      A,B,Ci
            outputs     S,Co
G1:         HalfAdder   (A,B),(Sum1,Carry1)
G2:         HalfAdder   (Sum1,Ci),(S,Carry2)
G3:         or          (Carry1,Carry2),Co
            endcircuit
```

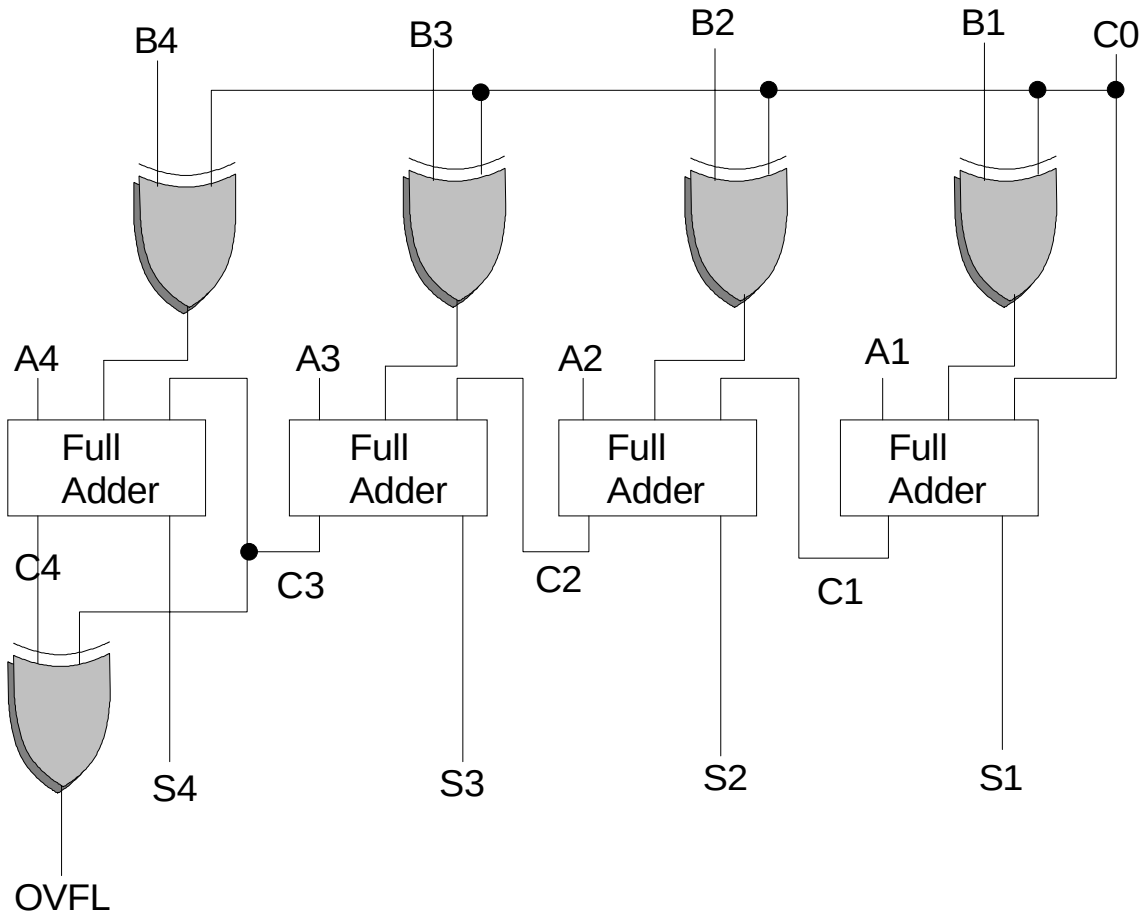
The previous definition for the half adder must follow this circuit in the same file. Enter this circuit and test it.

To construct a 16-bit adder, chain 16 full adders together as illustrated in the following diagram. This diagram illustrates only a 4-bit adder. Use your imagination to extend this to 16-bits.



You should be able to code this in FHDL with no problem Remember that the main circuit must come first, followed by any subcircuits. The order of the subcircuits is unimportant. Code this circuit and test it. You may wish to start with fewer than 16 bits and work up to it.

Now, convert the 16-bit adder to an adder subtractor. This is done as illustrated in the following diagram.



Make the changes to the circuit and test it.

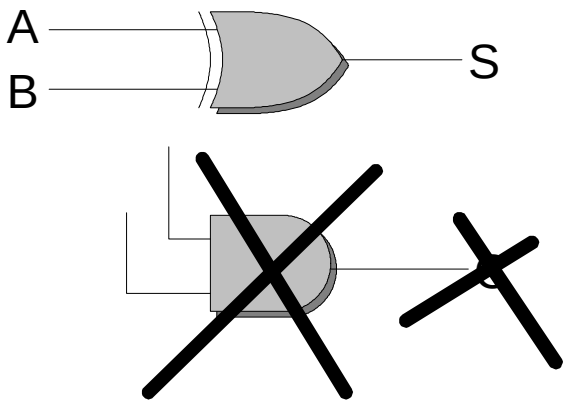
Results: Turn in a printout of the final circuit. Test the printout using at least 30 different inputs, and print a result of the trace. Show additions and subtractions. Show a carry from the low order bit position to the high order bit. (Subtract a **-1** from **0**.)

Experiment 2
Carry Lookahead Adders

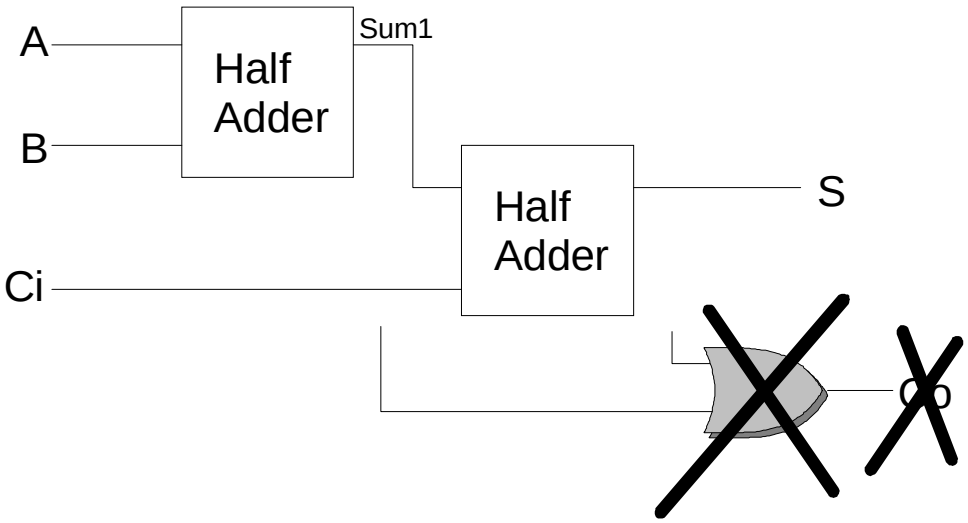
Objective: Learn the principles of designing hierarchical circuits, and the principles of carry lookahead addition..

Project: In this experiment, you will create a 16-bit carry lookahead adder/subtractor. This circuit will be based on the ripple-carry adder designed in **Experiment 1**.

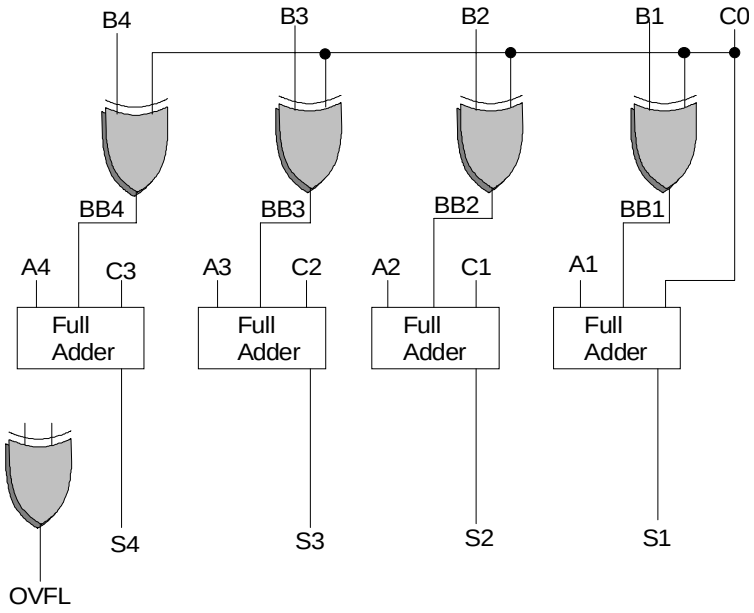
Procedure: Remove the carry logic from the half adder as illustrated in the following diagram.



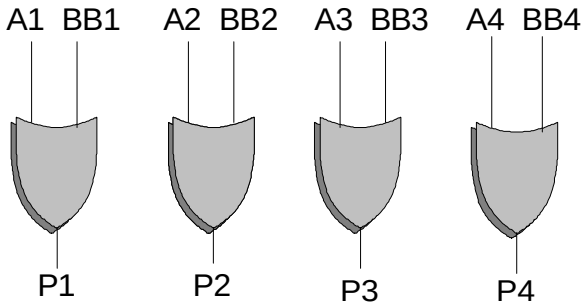
Remove the carry computation from the full adder as illustrated below.

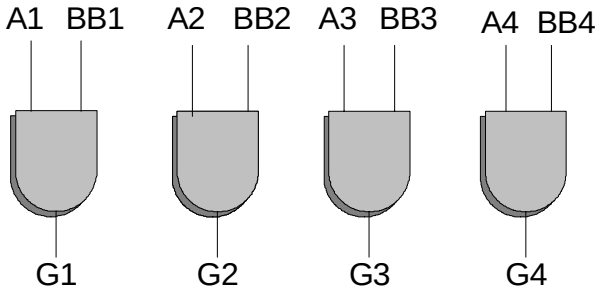


Take the ripple carry adder, and break the links as illustrated below.



The next step is to create the components needed by the carry lookahead adder. First create four-bit propagate and generate units, as illustrated below. The **BB** inputs will eventually come from the outputs of the **XOR** gates in the preceding diagram.





Now design a 4-bit carry lookahead unit using the following boolean formulas.

C1=G1+P1•C0
C2= G2+P2•G1+P2•P1•C0
C3= G3+P3•G2+P3•P2•G1+P3•P2•P1•C0
GG= G4+P4•G3+P4•P3•G3+P4•P3•P2•G1
GP= P4•P3•P2•P1

Creation of the carry lookahead circuitry will require five carry lookahead units. The following table lists the inputs and outputs of each.

Unit	Inputs	Outputs
1	P1,P2,P3,P4,G1,G2,G3,G4,C0	C1,C2,C3,GG1,GP1
2	P5,P6,P7,P8,G5,G6,G7,G8,C4	C5,C6,C7,GG2,GP2
3	P9,P10,P11,P12,G9,G10,G11,G12,C8	C9,C10,C11,GG3,GP3
4	P13,P14,P15,P16,G13,G14,G15,G16,C12	C13,C14,C15,GG4,GP4
5	GP1,GP2,GP3,GP4,GG1,GG2,GG3,GG4,C0	C4,C8,C12,GGP1,GGG1

The overflow computation will require computation of **C16**, which is done using the following equation.

C1=GGG1+GGP1•C0

Results: Turn in a printout of the final circuit. Test the printout using at least 30 different inputs, and print a result of the trace. Show additions and subtractions. Show a carry from the low order bit position to the high order bit. (Subtract a **-1** from **0**.)