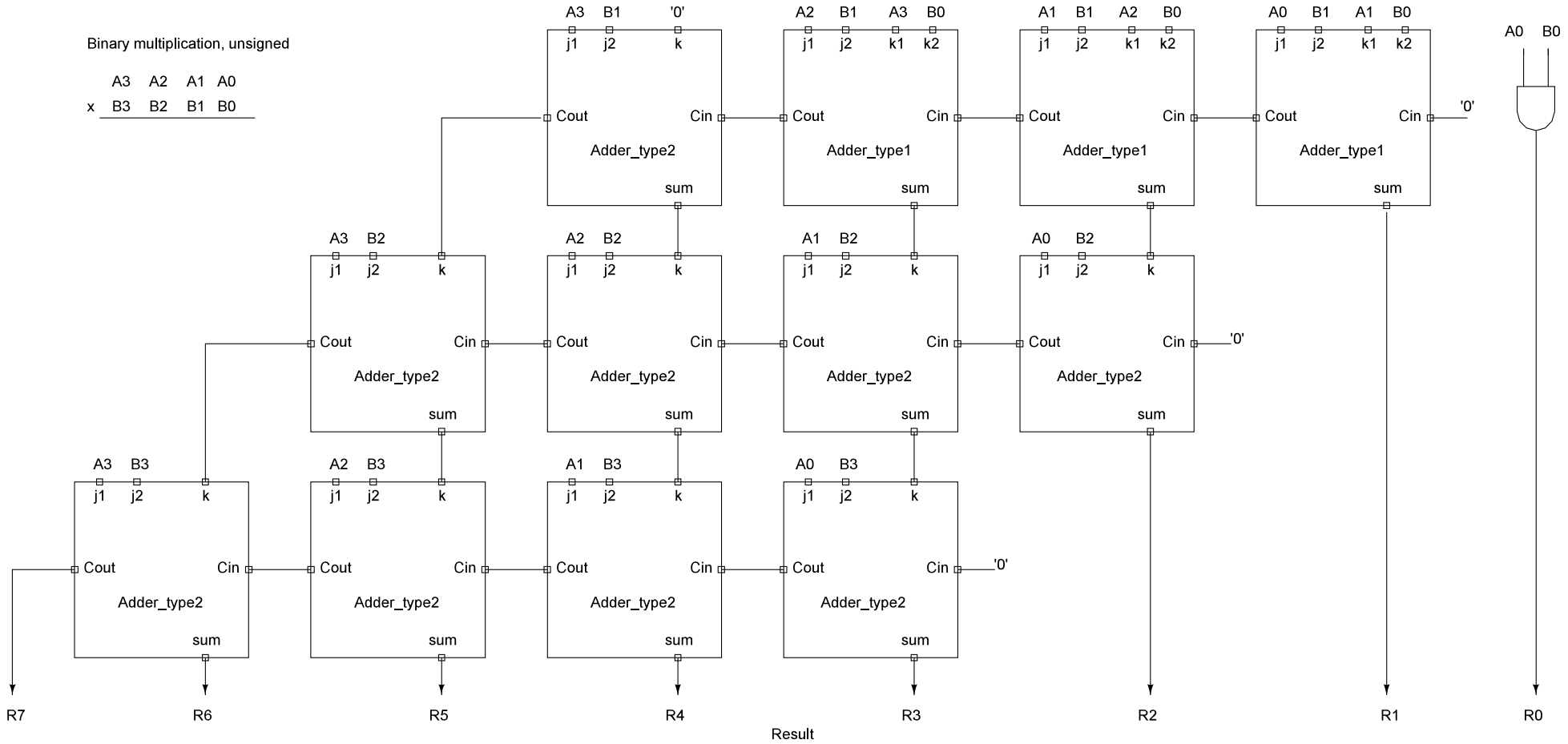
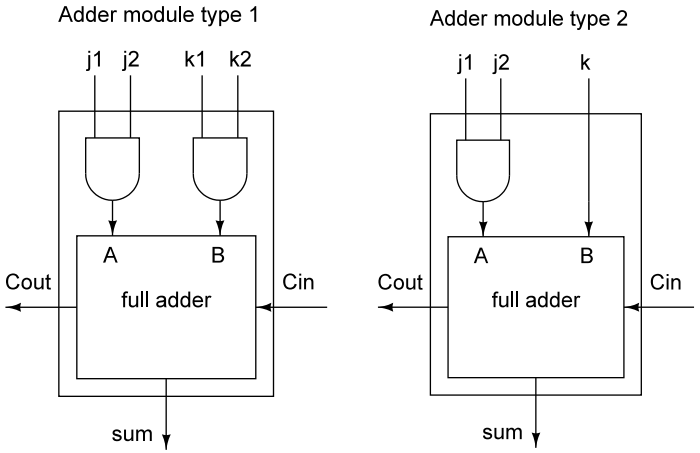

Binary multiplication, unsigned
A3 A2 A1 A0
x B3 B2 B1 B0
A3 B1 '0'
A2 B1 A3 B0
A1 B1 A2 B0
A0 B1 A1 B0
A0 B0
j1 j2 k
j1 j2 k1 k2
Cout
Cin
Adder_type2
Adder_type1
sum
'0'
A3 B2
A2 B2
A1 B2
A0 B2
A3 B3
A2 B3
A1 B3
A0 B3
R7
R6
R5
R4
R3
R2
R1
R0
Result


Adder module type 1
j1 j2 k1 k2
A B
Cout
Cin
full adder
sum
Adder module type 2
j1 j2 k
A B
Cout
Cin
full adder
sum

```
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adder_type1 is
    port (j1,j2,k1,k2,Cin : in std_logic;
          Cout, sum : out std_logic);
end adder_type1;

architecture adder1 of adder_type1 is

begin

end adder1;
----------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adder_type2 is
   (more statements go here)

----------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity lab4_solution_top is
    Port ( A0 : in  STD_LOGIC;
           A1 : in  STD_LOGIC;
           A2 : in  STD_LOGIC;
           A3 : in  STD_LOGIC;
           B0 : in  STD_LOGIC;
           B1 : in  STD_LOGIC;
           B2 : in  STD_LOGIC;
           B3 : in  STD_LOGIC;
           Led : out  STD_LOGIC_VECTOR (7 downto 0));
end lab4_solution_top;

architecture Behavioral of lab4_solution_top is

   component adder_type1
      port (j1,j2,k1,k2,Cin :in std_logic;
            Cout, sum : out std_logic);
   end component;

   component adder_type2
      port (j1,j2,k,Cin :in std_logic;
            Cout, sum : out std_logic);
   end component;

   signal .... : std_logic;

begin
   add1 : adder_type1 port map(A0,B1,A1,B0,'0',c1,Led(1));

end Behavioral;
```

The VHDL shell for lab 4. Note that two components are defined (adder_type1 and adder_type2) in the same file as the multiplier, i.e. lab4_solution. The component statements in the architecture of the multiplier description are needed as shown. The component descriptions could be done in a separate file or two separate files if desired. Because they are quite short, including them with the main description is good.

Note that the actual circuit descriptions of the adder circuits is left to the reader of this note to figure out. Refer to the preceding page and the block diagrams

In the lab4_solution_top architecture one copy of an adder is shown. Follow the block diagram to define the rest of the adders and their interconnection. Use a constraints file to make setting up the input/output ports easy..

A few other things to note:

1) The library IEEE statements are needed prior to each instance of an entity, as shown.
2) The order of the signal names listed in the port map of the instantiated adder_type1 (named add1) is the order that signals are defined in the adder_type1 entity, specifically j1, j2, k1, k2, Cin, Cout, sum. That means that incoming signal A0 connects to j1 in the adder, B1 to j2, etc.
3) Each signal that goes between blocks in the block diagram, such as Cout going to Cin needs to have a unique name and be defined in the architecture prior to the begin statement
4) Names defined in the entity of a component, such as adder_type1 or adder_type2, or within their architecture, are local to the component and not visible from the lab4_solution_top architecture.