Assignment #3
EECS 303: Advanced Digital Logic Design

Problem 1. Arithmetic Circuits (20 Points)

a. Assume that inverter, AND, and OR gates have 1ns delay and XOR, XNOR gates have 3ns delay. What is the worst case delay from any input to any sum output for the 4-bit carry lookahead adder shown in the figure below. What is the worst case delay from any any input to the carry output? The structure of the CLA is as discussed in class. The figures below should help to guide you.

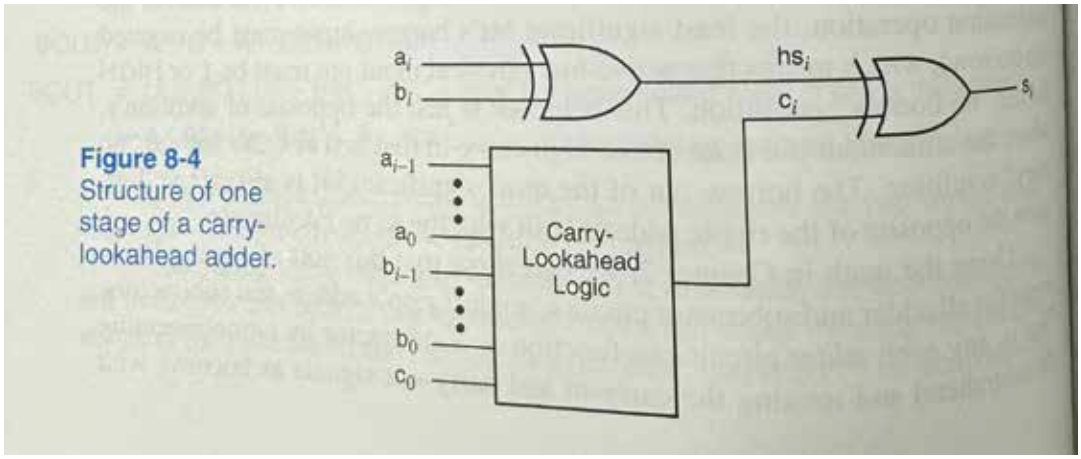
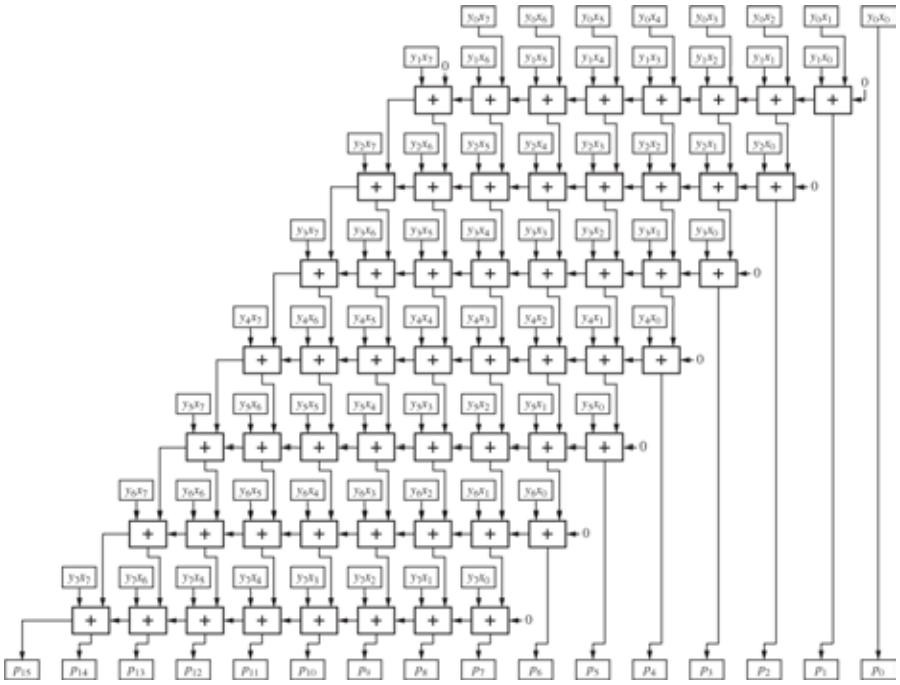


Figure 8-4
Structure of one stage of a carry-lookahead adder.

$$
\begin{aligned}
c_1 &= g_0 + p_0 \cdot c_0 \\
c_2 &= g_1 + p_1 \cdot c_1 \\
&= g_1 + p_1 \cdot (g_0 + p_0 \cdot c_0) \\
&= g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0 \\
c_3 &= g_2 + p_2 \cdot c_2 \\
&= g_2 + p_2 \cdot (g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0) \\
&= g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0 \\
c_4 &= g_3 + p_3 \cdot c_3 \\
&= g_3 + p_3 \cdot (g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0) \\
&= g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0
\end{aligned}
$$

b. Determine the worst-case propagation delay of the multiplier in Figure below, assuming that the propagation delay from any full-adder input to its sum output is twice as long as the delay to the carry output. Repeat, assuming the opposite relationship. If you were designing the adder cell from scratch, which path would you favor with the shortest delay?



Interconnections for an 8 × 8 combinational multiplier.

Problem 2. Arithmetic Circuits (10 Points)

What is the delay through the critical path in the 8-bit Kogge-Stone adder? Assume that it gets implemented with 2-input simple logic gates (and, or, exor, not) and that a gate delay equals 1 unit of time. Trace the path through the adder by listing the number and type of each gate.

Problem 3. Arithmetic Circuits (30 Points)

Write RTL-style Verilog module for a 3x3 (i.e., each input is 3 bits) combinational multiplier and simulate by using the following 16 input combinations.

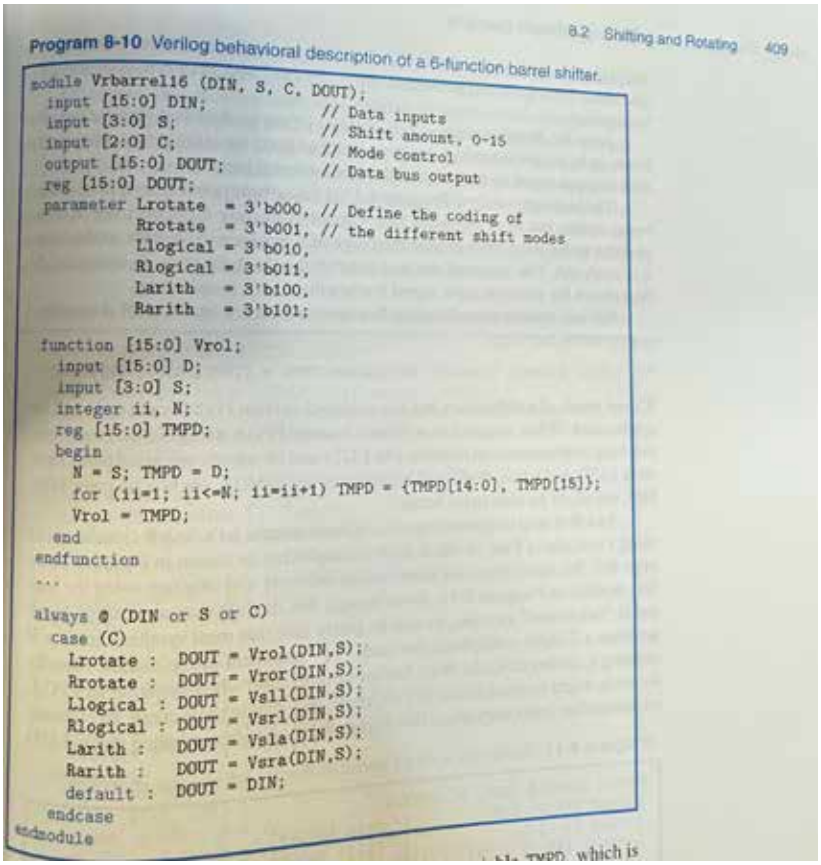
Input 1 = 5, Inout 2 = all possible value combinations
Input 1 = all possible value combinations, Input 2= 2

Use explicit **assign** statements to define each partial product calculation and you can use **assign** statements with arithmetic addition to accumulate those partial products.

Problem 4. Shifter Functions in Verilog (40 Points)

The table below lists all shift functions available in a barrel shifter. The Verilog code provides a partial template for a barrel shifter illustrating how one of the shift functions (Vrol) can be implemented in Verilog and embedded into the barrel shifter. Following this example complete the barrel shifter implementation by adding the functions for the remaining shift operations. Simulate all options of the barrel shifter by applying
DIN = [1001011101010011]
S = [0011]

Shift Type	*Name*	*Code*	*Function*	*Note*
Left rotate	Lrotate	000	Vrol	Wrap-around
Right rotate	Rrotate	001	Vror	Wrap-around
Left logical	Llogical	010	Vsll	0 into LSB
Right logical	Rlogical	011	Vslr	0 into MSB
Left arithmetic	Larith	100	Vsla	0 into LSB
Right arithmetic	Rarith	101	Vsra	Replicate MSB



8.2 Shifting and Rotating 409

Program 8-10 Verilog behavioral description of a 6-function barrel shifter.

```
module Vrbarrel16 (DIN, S, C, DOUT);
  input [15:0] DIN;              // Data inputs
  input [3:0] S;                 // Shift amount, 0-15
  input [2:0] C;                 // Mode control
  output [15:0] DOUT;            // Data bus output
  reg [15:0] DOUT;
  parameter Lrotate  = 3'b000, // Define the coding of
            Rrotate  = 3'b001, // the different shift modes
            Llogical = 3'b010,
            Rlogical = 3'b011,
            Larith   = 3'b100,
            Rarith   = 3'b101;

  function [15:0] Vrol;
    input [15:0] D;
    input [3:0] S;
    integer ii, N;
    reg [15:0] TMPD;
    begin
      N = S; TMPD = D;
      for (ii=1; ii<=N; ii=ii+1) TMPD = {TMPD[14:0], TMPD[15]};
      Vrol = TMPD;
    end
  endfunction
  ...
  always @ (DIN or S or C)
    case (C)
      Lrotate :  DOUT = Vrol(DIN,S);
      Rrotate :  DOUT = Vror(DIN,S);
      Llogical : DOUT = Vsll(DIN,S);
      Rlogical : DOUT = Vsrl(DIN,S);
      Larith :   DOUT = Vsla(DIN,S);
      Rarith :   DOUT = Vsra(DIN,S);
      default :  DOUT = DIN;
    endcase
endmodule
```

... TMPD, which is