Problem 1) With dataflow Verilog, describe the Generate/Propagate Unit, the Carry-Lookahead Unit, and the Summation Unit in Figure 1 as separate modules. Do not include delays in your models. We will add them later in the lab experiments. Use the module interfaces below as a guide. Gate-level schematics can be hard to read so you may find expressions (3) through (11) easier to follow.

```
***This Generate-Propagate Unit for 4-bit carry-lookahead adder
 * Computes the generate (G) and propagate (P) signals for each bit.
 */

module generate_propagate_unit (G, P, X, Y);
 output wire [3:0] G, P; // 4-bit Generate and Propagate outputs
 input wire [3:0] X, Y; // 4-bit inputs (X and Y)

 // Compute Generate and Propagate signals for each bit
 assign G = X & Y;     // Generate: gi = xi * yi
assign P = X ^ Y;     // Propagate: pi = xi XOR yi
endmodule

/////////
* Carry Lookahead Unit for 4-bit carry-lookahead adder
* Computes the carry signals for each bit position based on G, P, and input carry.
 //////////

module carry_lookahead_unit (C, G, P, C0);
 output wire [4:1] C;  // Carry outputs (C1, C2, C3, C4)
 input wire [3:0] G, P; // 4-bit Generate and Propagate inputs
 input wire C0;  // Initial carry input (usually 0 or previous carry)

 // This Compute carry signals for each bit position

 assign C[1] = G[0] | (P[0] & C0);
assign C[2] = G[1] | (P[1] & G[0]) | (P[1] & P[0] & C0);
assign C[3] = G[2] | (P[2] & G[1]) | (P[2] & P[1] & G[0]) | (P[2] & P[1] & P[0] & C0);
assign C[4] = G[3] | (P[3] & G[2]) | (P[3] & P[2] & G[1]) | (P[3] & P[2] & P[1] & G[0]) | (P[3] & P[2] &
P[1] & P[0] & C0);

endmodule
```

```
/* * Summation Unit for 4-bit carry-lookahead adder
* Computes the sum bits based on propagate signals and carry bits.
*/

module summation_unit (S, P, C);
 output wire [3:0] S;// 4-bit Sum output
 input wire [3:0] P, C; // 4-bit Propagate and Carry inputs

 assign S = P ^ C // Shifted carry inputs to match each bit position

 endmodule
```

Question 2) Now, use structural Verilog along with the modules you have just created to wire up a 4-bit Carry Lookahead adder. The module interface you should use is provided below.

```
// Top-level module for a 4-bit Carry-Lookahead Adder (CLA)
 module carry_lookahead_4bit(Cout, S, X, Y, Cin);

 // Output ports
 output wire Cout; // Carry-out of the 4-bit adder
 output wire [3:0] S; // 4-bit sum output

 // Input ports
 input wire [3:0] X, Y; // 4-bit input operands
 input wire Cin; // Input carry

 // Internal wires for generate and propagate signals
 wire [3:0] G, P; // Generate (G) and Propagate (P) signals for each bit
 wire [4:1] C; // Carry signals between each bit stage (C[4] is final carry)

 // Module instantiations
 generate_propagate_unit GPU (.G(G), .P(P), .X(X), .Y(Y)); // Generate and propagate signals
 carry_lookahead_unit CLA (.C(C), .G(G), .P(P), .C0(Cin)); // Carry lookahead logic
 summation_unit SU (.S(S), .P(P), .C(C[3:0])); // Summation logic for each bit

 // Final carry-out assignment
 assign Cout = C[4]; // Connect final carry to the output
 endmodule
```

Question 3) What is the gate-count of your 4-bit carry-lookahead adder?

The gate count is 26

Question 4) The previous problems were concerned with a single-level 4-bit carry-lookahead adder. In one of the lab experiments, we will construct a 16-bit, 2-level carry-lookahead adder. The following questions will prepare you for this exercise. What is the propagation delay of the 16-bit, 2-level carry-lookahead adder in Figure 2? Likewise, what is the gate-count?

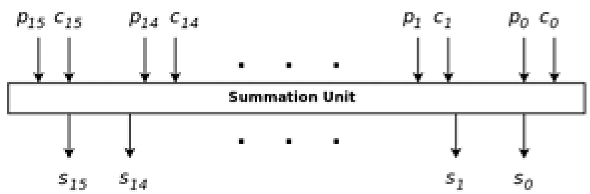

Figure 2: Two-Level Carry-Lookahead Adder

The propagation delay of the 16-bit, two-level carry-lookahead adder is 6 gate delays.

The total gate count is calculated as follows:

$32+14\times4+14\times4+16=188$ gates