Prelab Questions

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 modules. We will add them later in the lab experiments.**

Code Block 1: Generate Propagate Unit

```
1  // This module describes the Carry Generate/Propaget
2  // Unit for a 4-bit carry-lookahead addition
3  module generate_propagate_unit(G, P, X, Y);
4
5    // Ports are wires as we will use dataflow
6    output wire [3:0] G, P;
7    input wire [3:0] X, Y;
8
9    assign G[0] = X[0] & Y[0];
10   assign G[1] = X[1] & Y[1];
11   assign G[2] = X[2] & Y[2];
12   assign G[3] = X[3] & Y[3];
13
14   assign P[0] = X[0] ^ Y[0];
15   assign P[1] = X[1] ^ Y[1];
16   assign P[2] = X[2] ^ Y[2];
17   assign P[3] = X[3] ^ Y[3];
18
19 endmodule // generate_propagate_unit
```

Code Block 2: Carry Lookahead Unit

```
1  // This module describes the 4-bit carry-lookahead Unit
2  // for a carry-lookahead addaer
3  module carry_lookahead_unit (C, G, P, C0);
```

```
4    // Ports are wires because we will use dataflow
5    output wire [4:1] C; // C4, C3, C2, C1
6    input wire [3:0] G, P; // Generates and propagates
7    input wire C0; // input carry
8
9    assign C[1] = G[0] || (P[0] & C0);
10   assign C[2] = G[1] || (P[1] & G[0]) || (P[1] & P[0] & C[0]);
11   assign C[3] = G[2] || (P[2] & G[1]) || (P[2] & P[1] & G[0)
12     || (P[2] & P[1] & P[0] & C[0]);
13   assign C[4] = G[3] || (P[3] & G[2]) || (P[3] & P[2] & G[1])
14     || (P[3] & P[2] & P[1] & G[0) || (P[3] & P[2] & P[1] &
15    P[0] & C[0]);
16
17 endmodule // carry_lookahead_unit
```

Code Block 3: Summation Unit

```
1  // This module describes the 4-bit summation Unit
2  // for a carry-lookahead adder
3
4  module summation_unit (S, P, C);
5
6    // Ports are wires becasue we will use dataflow
7    output wire [3:0] S; // sum vector
8    input wire [3:0] P, C;
9
10   assign  S[0] = P[1] ^ C[1];
11   assign  S[2] = P[2] ^ C[2];
12   assign  S[3] = P[3] ^ C[3];
13   assign  S[4] = P[4] ^ C[4];
14
15 endmodule // summation_unit
```

2. **Now, use structural Verilog along with the modules you have just created to wire up a 4-bit Carry-Lookahead adder.**

Code Block 4: Carry Lookahead 4-Bit

```
1  // This is the top-level module for a 4-bit
2  // carry-lookahead addaer
3
4  module carry-lookahead_4bit (Cout, S, X, Y, Cin);
5
6    // Ports are wires because we are using structual
7    output wire Cout; // C4 for a 4-bit adder
8    output wire [3:0] S; // final 4-bit sum vector
9    input wire [3:0] X, Y; // the 4-bit addends
10   input wire Cin; // input carry
11
12   // Intermediete wires
13   wire [3:0] G, P;
14   wire [4:1] C;
15
16   // generate_propagate_unit(G, P, X, Y);
17   generate_propagate_unit gpu(G, P, X, Y);
18
19   // carry_lookahead_unit (C, G, P, C0);
20   carry_lookahead_unit clu(C, G, P, Cin);
21
22   // summation_unit (S, P, C);
23   summation_unit su(S, P, C);
24
25   assign Cout = C[4];
26 endmodule // carry-lookahead_4bit
```

3. **What is the gate count of your 4-bit *carry-lookahead* adder?**

   The gate count is 26.

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?**

   The propagation delay of the 16-bit is 6. The gate count is calculated below

$$32 + 14 \cdot 4 + 14 + 16 = 188 \text{ gates}$$