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

```verilog
// This module describes the Carry Generate/Propaget
// Unit for a 4-bit carry-lookahead addition
module generate_propagate_unit(G, P, X, Y);


  // Ports are wires as we will use dataflow
  output wire [3:0] G, P;
  input wire [3:0] X, Y;

  assign G[0] = X[0] && Y[0];
  assign G[1] = X[1] && Y[1];
  assign G[2] = X[2] && Y[2];
  assign G[3] = X[3] && Y[3];

  assign P[0] = X[0] ^ Y[0];
  assign P[1] = X[1] ^ Y[1];
  assign P[2] = X[2] ^ Y[2];
  assign P[3] = X[3] ^ Y[3];

endmodule // generate_propagate_unit
```

Code Block 2: Carry Lookahead Unit

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

```
  // Ports are wires because we will use dataflow
  output wire [4:1] C; // C4, C3, C2, C1
  input wire [3:0] G, P; // Generates and propagates
  input wire C0; // input carry

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

endmodule // carry_lookahead_unit
```

Code Block 3: Summation Unit

```
// This module describes the 4-bit summation Unit
// for a carry-lookahead adder

module summation_unit (S, P, C);

  // Ports are wires becasue we will use dataflow
  output wire [3:0] S; // sum vector
  input wire [3:0] P, C;

  assign  S[0] = P[1] ^ C[1];
  assign  S[2] = P[2] ^ C[2];
  assign  S[3] = P[3] ^ C[3];
  assign  S[4] = P[4] ^ C[4];

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

```
// This is the top-level module for a 4-bit
// carry-lookahead addaer

module carry-lookahead_4bit (Cout, S, X, Y, Cin);

  // Ports are wires because we are using structual
  output wire Cout; // C4 for a 4-bit adder
  output wire [3:0] S; // final 4-bit sum vector
  input wire [3:0] X, Y; // the 4-bit addends
  input wire Cin; // input carry

  // Intermediete wires
  wire [3:0] G, P;
  wire [3:1] C;

  // generate_propagate_unit(G, P, X, Y);
  generate_propagate_unit gpu0(G[0], P[0], X[0], Y[0]);
  generate_propagate_unit gpu1(G[1], P[1], X[1], Y[1]);
  generate_propagate_unit gpu2(G[2], P[2], X[2], Y[2]);
  generate_propagate_unit gpu3(G[3], P[3], X[3], Y[3]);

  // carry_lookahead_unit (C, G, P, C0);
  carry_lookahead_unit clu0(C[1], G[0], P[0], Cin);
  carry_lookahead_unit clu1(C[2], G[1], P[1], C[1]);
  carry_lookahead_unit clu2(C[3], G[2], P[2], C[2]);
  carry_lookahead_unit clu3(Cout, G[3], P[3], C[3]);

  // summation_unit (S, P, C);
  summation_unit su0(S[0], P[0], C[0]);
  summation_unit su1(S[1], P[1], C[1]);
  summation_unit su2(S[2], P[2], C[2]);
  summation_unit su3(S[3], P[3], C[3]);

endmodule // carry-lookahead_4bit
```

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

Module	# of Gates
gpu0	8
gpu01	8
gpu2	8
gpu3	8
clu0	8
clu1	8
clu2	8
clu3	8
su0	4
su1	4
su2	4
su3	4
Total	**80**

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?**