HDL Design Homework: 4-Bit Adder Architectures

Learning Objectives

By completing this assignment, you will:

- Implement arithmetic circuits using structural and dataflow HDL modeling styles
- Quantify performance differences between ripple-carry and carry-lookahead architectures
- Correlate HDL simulation waveforms with circuit-level timing behavior
- Analyze the area-speed tradeoff fundamental to digital design

Part 1: Design Specifications (40 points)

A. Ripple-Carry Adder (RCA)

Design a 4-bit ripple-carry adder using structural Verilog/VHDL:

- Instantiate four 1-bit full adders as separate modules/components
- Full adder specification (Verilog example):

```
module full_adder(input a, b, cin, output s, cout);
  assign s = a ^ b ^ cin;
  assign cout = (a & b) | (b & cin) | (a & cin);
endmodule
```

- Top-level ports:

```
input [3:0] A, B
input Cin
output [3:0] S
output Cout
```

B. Carry-Lookahead Adder (CLA)

Design a **4-bit carry-lookahead adder** using **dataflow modeling**:

- Compute generate (G_i = A_i & B_i) and propagate (P_i = A_i ^ B_i) signals
- Compute carries in parallel:

C1 = G0 | (P0 & Cin)

C2 = G1 | (P1 & G0) | (P1 & P0 & Cin)

C3 = G2 | (P2 & G1) | (P2 & P1 & G0) | (P2 & P1 & P0 & Cin)

C4 = G3 | (P3 & G2) | (P3 & P2 & P1 & G0) | (P3 & P2 & P1 & P0 & Cin)

- Top-level ports identical to RCA for fair comparison

Part 2: Testbench & Simulation (30 points)

Create a **single parameterized testbench** that:

1. Instantiates **both adder architectures** with identical inputs
2. Applies these test vectors (toggle inputs every 10 ns):

Cycle	A (binary)	B (binary)	Cin	Purpose
1	0000	0000	0	Zero case
2	1111	0001	0	Ripple through all bits
3	1010	0101	1	Mixed propagation
4	1111	1111	1	Maximum carry generation
5	0111	0001	0	**Critical path test**

3. Simulates for ≥100 ns with <100 ps rise/fall times

Required waveforms to submit:

• **Figure 1**: RCA vs. CLA sum outputs (S_rca[3:0], S_cla[3:0]) and final carry (Cout_rca, Cout_cla)
• **Figure 2**: Critical path zoom-in (5–25 ns window) showing:
– RCA: Sequential Cin → C1 → C2 → C3 → C4 transitions
– CLA: Simultaneous carry generation
• Annotate both figures with **propagation delays** (ns) measured from input change to stable output

---

Part 3: Analysis & Discussion (30 points)

Answer concisely (≤300 words total):

1. **Timing comparison**
   From your waveforms, measure delay from Cin change to stable Cout for test vector #5. Report values (ns). Explain why RCA exhibits sequential carry propagation while CLA does not.
2. **Circuit explanation**
   Explain *at the gate level* why RCA's worst-case delay scales as **O(n)** for n bits while CLA scales as **O(log n)**. Reference your generate/propagate equations.
3. **Area-speed tradeoff**
   Estimate gate count difference between 4-bit RCA and CLA. Why might designers still choose RCA for wide adders (>16 bits)? When would CLA be preferred?
4. **Real-world relevance**
   Modern FPGAs implement adders using dedicated carry chains. How does this hardware feature mitigate RCA's delay disadvantage? (2–3 sentences)

Submission Requirements

Submit a **single PDF** containing:
☐ Complete, synthesizable HDL code for both adders + testbench (with comments)
☐ Two annotated waveform figures as specified in Part 2
☐ Typed responses to all Part 3 questions
☐ **Optional**: Synthesis report snippets showing LUT/FF usage difference (extra credit)

File naming: LastName_HW3.pdf
Late policy: -20% per day

Design Tips

- Use $monitor or waveform cursors to measure precise delays
- For CLA, declare intermediate G[3:0] and P[3:0] wires for readability
- Simulate with **20 ps time precision** (timescale 1ns/20ps)
- Verify functional correctness first—both adders must produce identical results!

References

- Harris & Harris, *Digital Design and Computer Architecture*, Ch. 5.1–5.2
- IEEE Std 1364-2005 (Verilog) or IEEE Std 1076-2019 (VHDL)
- Xilinx UG474: "7 Series FPGAs Carry Logic Resources"

"The carry-lookahead adder exemplifies a fundamental VLSI principle: reducing critical path depth at the cost of increased wiring complexity."
— Rabaey et al., *Digital Integrated Circuits*