

Lab 6

Adder Circuits and Latency

Overview

Section 1: Implementations of ***25-bit Adders*** and timing analysis in LogicWorks

1. Ripple Adder
2. Single-Level Carry Look-Ahead (CLA) Adder
3. Two-Level CLA Adder

Section 2: Implementation of two synchronous ***25-bit Adders*** and timing analysis in Verilog

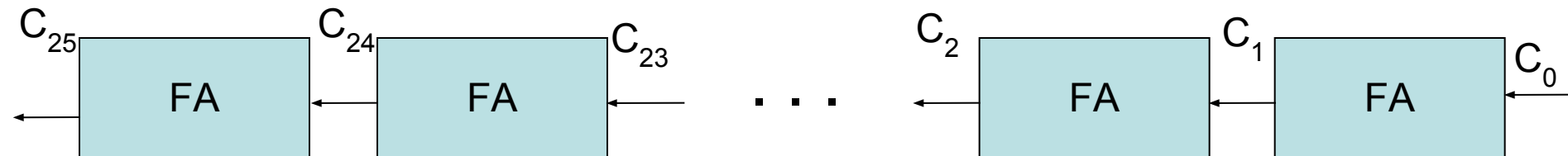
Ripple Adder

- Idea

- Daisy chain together as many full adders as needed for a simple, regular circuit implementation

- Component

- single-bit full adder for every bit



Carry Look-Ahead Adder

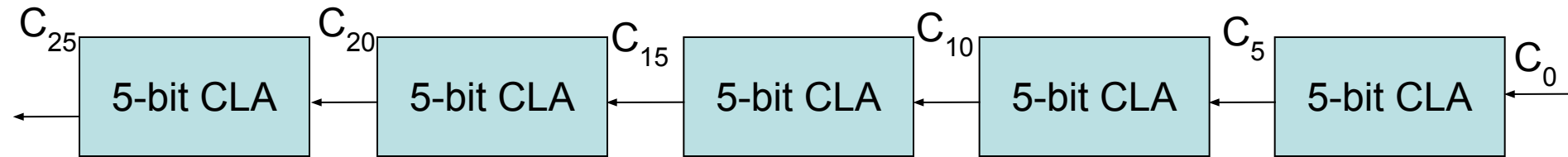
- Idea
 - Reduce the delay of the carry chain by precomputing the carry value for each bit
- Components
 - 1-bit half adder for each bit
 - Carry-generate function and carry-propagate function for each bit
 - Carry look-ahead logic

Carry-Generate & Carry-Propagate

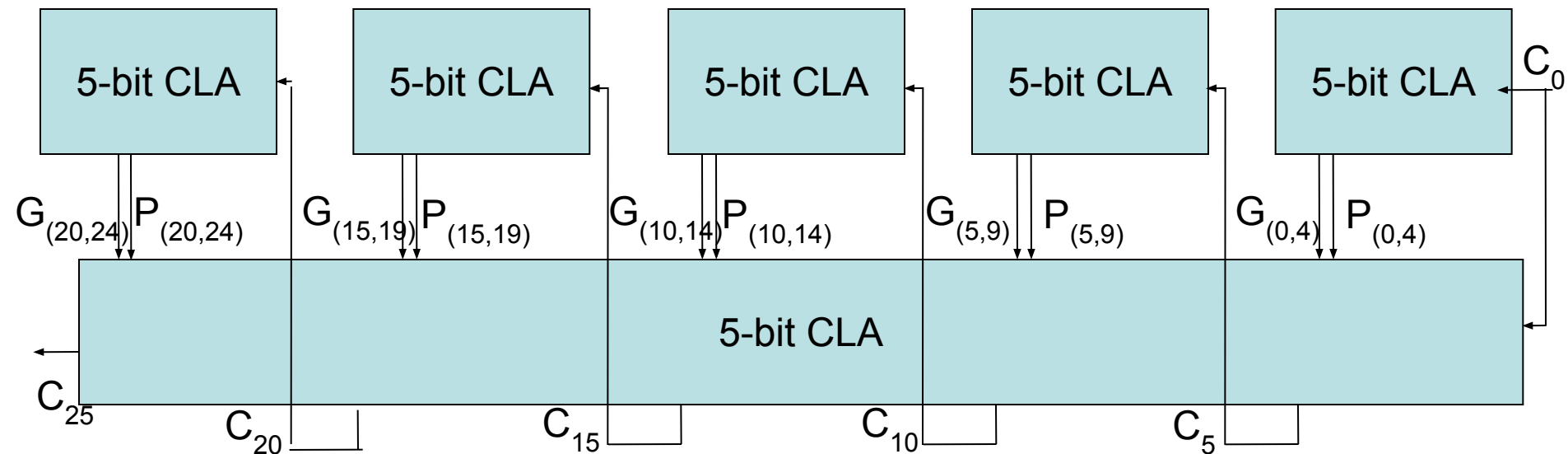
- **Carry-generate function g_i**
 - $g_i = x_i y_i$
 - If $g_i = 1$, a carry-out will definitely happen
 - Determines whether the two input values will generate a carry to a higher bit
- **Carry-propagate function p_i**
 - $p_i = x_i \oplus y_i$
 - If $p_i = 1$, the carry-in from the lower bit will be propagated to the carry-out
 - Determines whether the two input values can propagate a carry coming from a lower bit to a higher bit

25-bit CLA Adders

- Single-level CLA generator



- Two-level CLA generator



Implementation

- Implement a 5-bit carry look-ahead adder
- Measure the delays of different paths
- Design a single-level 25-bit carry look-ahead adder based on five 5-bit carry look-ahead adders
- Design a two-level 25-bit carry look-ahead adder based on five 5-bit carry look-ahead adders and one additional CLA generator

Implementation Requirements

- Reuse as many logic gates as possible
- Hierarchical design
 - Package single-bit Full Adder into a component
 - Package CLA generator into a component
 - Package 5-bit CLA binary adder into a component

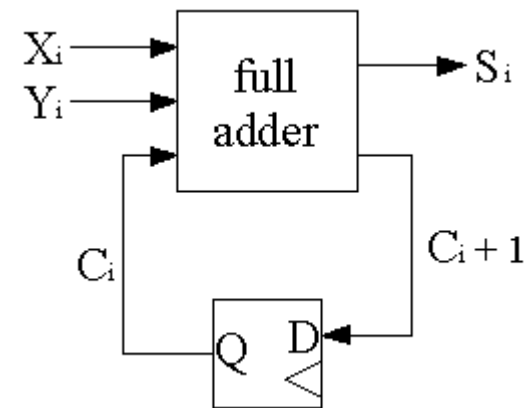
Synchronous Adder

■ Idea

- The single-bit FA is implemented as a FSM
- Computation is performed every clock cycle until the full sum is computed
- Hardware reuse
- Carries ripple through via the state of the FSM from cycle to cycle

■ Components

- 1 single-bit sequential full adder
inputs: x_i and y_i
output: s_i
- 1 5-bit sequential full adder
inputs: $x_i \dots x_{i+4}$ and $y_i \dots y_{i+4}$
output: $s_i \dots s_{i+4}$



Clock Period Considerations

Next State Logic

The combinational circuit for next state value's critical path is the lower bound on the clock period

Output Logic

The combinational circuit for outputs does not directly define the clock period but you must be careful that the outputs are available long enough for the external circuit

Implementation

- Implement a single-bit synchronous adder
- Implement the 25-bit adder using the single-bit component
- Calculate latency (Clock Period * Cycle Count)
- Implement a 5-bit synchronous adder
- Implement the 25-bit adder using the 5-bit component
- Again calculate latency (Clock Period * Cycle Count)

Implementation Guidelines

- Use bit shifting to input the correct values each cycle and avoid large muxes
- Use bit shifting to store outputs in their correct position and avoid large muxes
- You should make a 25-bit shifter component to make design simpler

Implementation Guidelines

- Ensure the delay values are in place between all statements
- Use only structural Verilog
- Do not modify top level design interface
- Use the provided reset framework