

Lab 4

Part 1: Look-ahead Subtractor

- Implementation of a *look-ahead subtractor* and delay analysis

Part 2: Static Hazard Analysis

- Hazard analysis and implementation of hazard-free circuits

Carry Look-ahead Adder

Idea

- Reduce the delay of the carry chain by precomputing the carry value for each bit

Components

- A 1-bit half adder for every bit
- A carry-generate function and carry-propagate function for every bit
- A carry look-ahead logic (for every 5 bits)

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
- **Carry-propagate function p_i**
 - $p_i = x_i \oplus y_i$
 - If $p_i = 1$, the carry-in from the next lower bit is propagated to the carry-out

Generating Your Carries

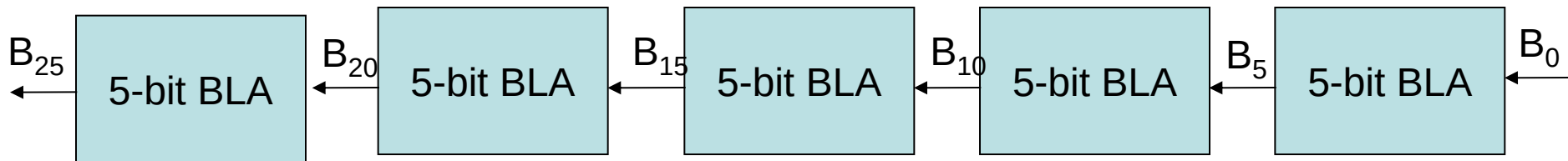
- General function:
 - $c_{i+1} = g_i + p_i * c_i$
- We can define subsequent carries in terms of c_0 :
 - $c_1 = g_0 + p_0 * c_0$
 - $c_2 = g_1 + p_1 * c_1 = g_1 + p_1 g_0 + p_1 p_0 c_0$
 - $c_3 = g_2 + p_2 * c_2 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$
 - $c_4 = ???$
 - $c_5 = ???$

Borrow Look-ahead Subtractor

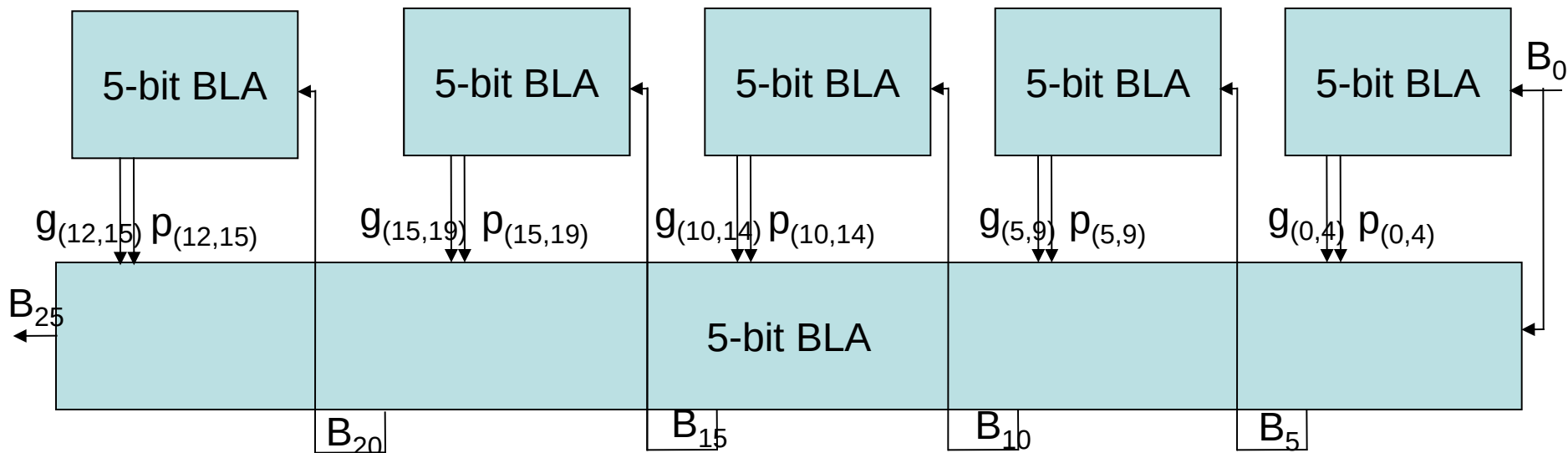
- Shares the same design principles with carry look-ahead adder
- **Borrow-generate function g_i**
 - If $g_i = 1$, a borrow-out will definitely happen
- **Borrow-propagate function p_i**
 - If $p_i = 1$, the borrow-in from the next lower bit is propagated to the borrow-out
- **Borrow look-ahead generator (BLA) logic**
 - Takes g_i , p_i and b_0 as inputs
 - Outputs the borrow value for each bit

4-digit (16-bit) BLA Generator

- Single-level BLA generator



- Two-level BLA generator



Implementation

- Single-bit subtractor: computes the difference along with the inverted borrow-generate and inverted borrow-propagate functions
- 5-bit BLA generator: computes borrows using generate and propagate values
- 5-bit subtractor: uses 5 single-bit subtractors and 1 BLA generator
- 25-bit subtractors: implement one-level and two-level versions

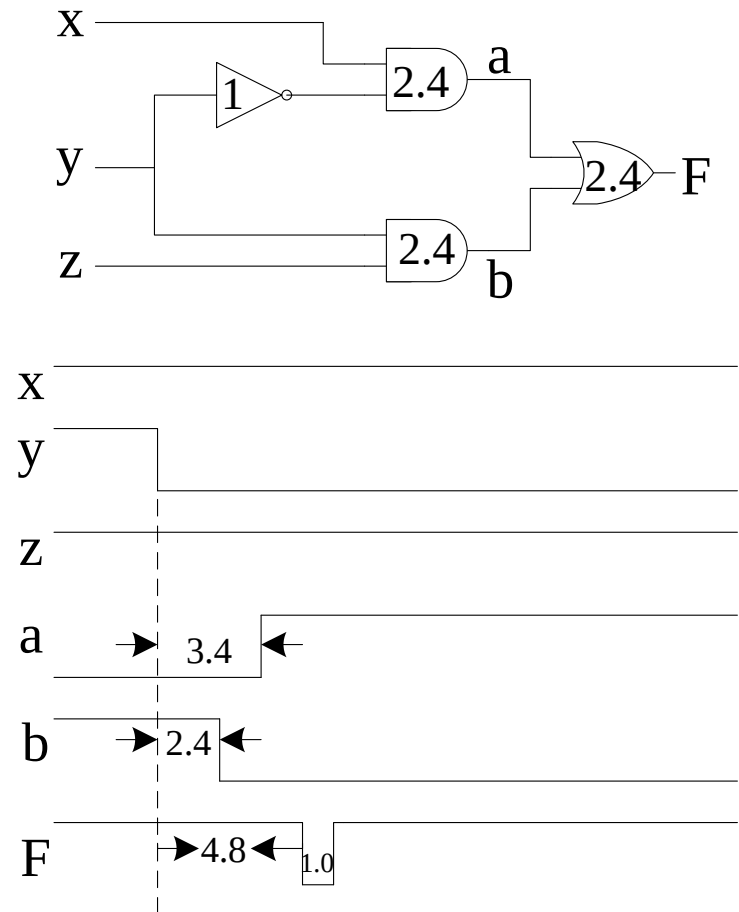
Implementation Requirements

- Modular design
 - Package the BLA generator into a module
 - Package the 1-bit binary subtractor into a module
 - Connect these to make a 5-bit subtractor
 - Connect 5 5-bit subtractors together to make a 25-bit subtractor
 - Add in another BLA generator for the second level
- Use structural Verilog to describe your circuits
- Remember that each gate you instantiate needs to have its delay specified

Static Hazards

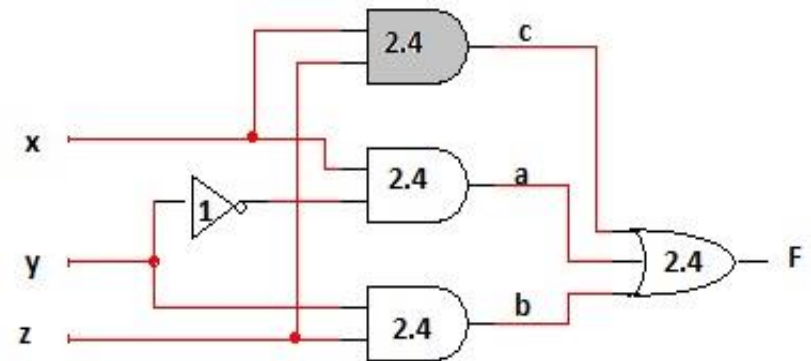
- **Effect:** short unwanted glitches at the output
- **Reason:** different propagation delays on converging paths

How do we fix it?



Static Hazards

- Extra gate as shown takes care of the hazard!



Minimal Hazard-free Circuit

- ***Transition Don't Cares:*** input transitions that will never occur during the regular operation of the circuit
- ***Reduce redundant hardware:*** remove the gates that were added for handling a static hazard if the transition that could have generated the static hazard is in the *Transition Don't Care* set.

Design 3 Circuits

- Circuit 1 and 2
 - Both the **Minimal** Implicant and the **Minimal** Implicate representations generate a hazard
- Circuit 3
 - Multiple naïve implementations
 - Only one makes the most use of the transition don't cares
- Remember that static hazards only occur in one direction!