NC State University
Department of Electrical and Computer Engineering

ECE 310 – Fall 2025

Lab 3

Summary

In this lab, you will create a dataflow model for a 4-bit Kogge-Stone adder (KSA).

Deliverables

1. Code for model and testbench as separate .v files uploaded to Moodle.
2. Short report describing the theory of operation, design/diagrams, snippets of code as needed, results, and a discussion. Include a screenshot of simulation waveforms for a 4-bit Kogge-Stone adder with 4 pairs of representative values. Justify why you used these values. Upload the report to Gradescope.
3. Extra Credit – up to 10 additional points will be added for typesetting your report in LaTeX, depending on usage of LaTeX elements (e.g. figures, tables, sections, etc)

Instructions

1. Create a new project in Vivado named **ksa_4bit_df**.
2. A Kogge-Stone adder is a type of fast adder, known as a prefix adder, that pre-computes the carries for faster addition. While not the most efficient implementation of a prefix adder, it offers several advantages over other fast adders, such as lower fan-out at each stage.
   Carries are pre-computed as generates (G) and propagates (P). An adder generates a carry when both inputs are High ($G = AB$) and it propagates an incoming carry when either input is High ($P = A \oplus B$). A version of this happens at every stage in the adder with different inputs. The diagram in Fig. 1 shows the operations performed at each stage of carry computation. This diagram only calculates the carries for the overall addition operation, the final Sum is computed via $S_i = P_i \oplus C_{i-1}$, where P_i is the original propagate for each bit position and C_{i-1} is the result of carry calculation for previous bit position, $C_i = G_i$ for final stage computation below.

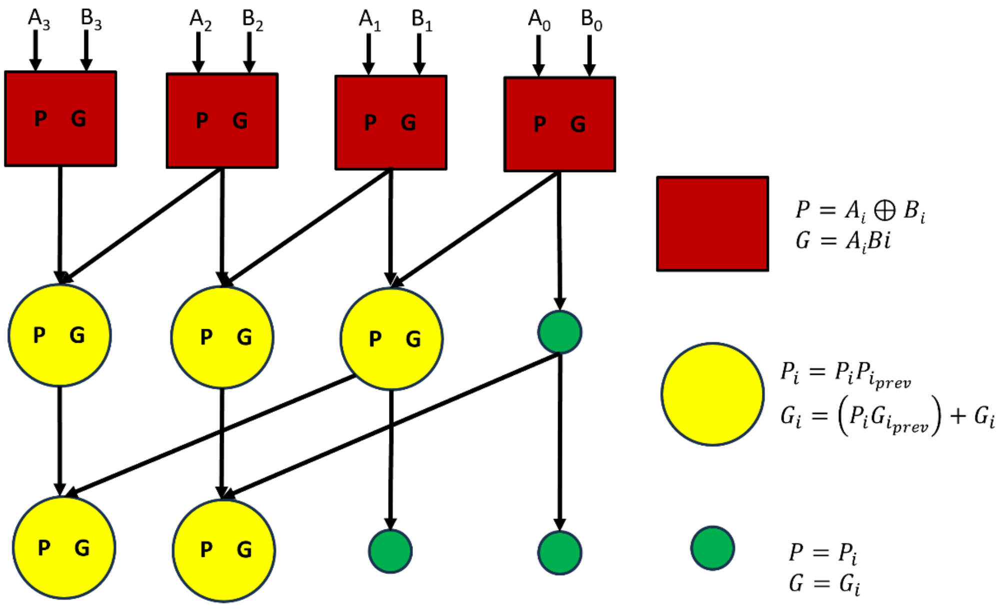

Fig. 1: Carry calculation in Kogge-Stone adder.

For more information on the operation of a Kogge-Stone adder, see the Wikipedia page: https://en.wikipedia.org/wiki/Kogge%E2%80%93Stone_adder

Do not implement the Carry-Lookahead Adder (CLA) equation, which is the most common solution found online. **Implement each stage of the operations in the KSA as a separate signal to be used as needed for later stages of calculation.** The KSA equation can be implemented as a one-line implementation, but it is not in the spirit of how a KSA operates and the advantages it provides, so use separate signals for each of the equations/blocks in the diagram in Fig. 1.

As an example run, addition of **A=4'b1010** and **B=4'b1101** to get **S=4'b0111** and **Cout=1'b1**, the values for each of the nodes in the tree is as shown in Fig. 2.

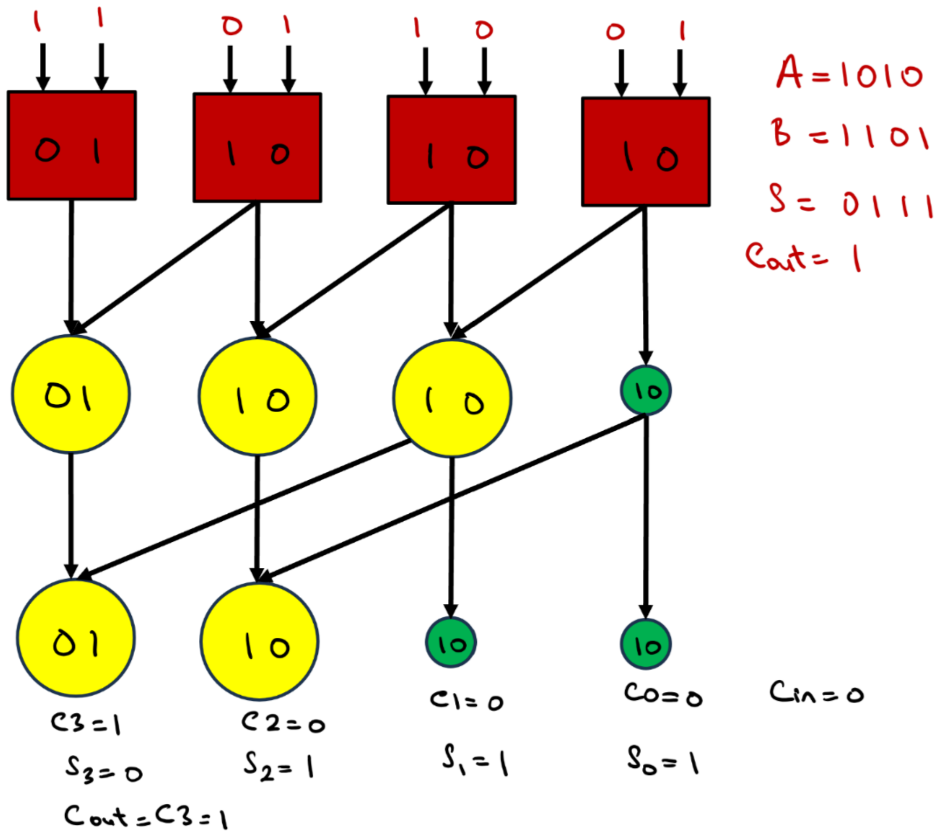

Fig. 2: Addition of **A=4'b1010** and **B=4'b1101** to get **S=4'b0111** and **Cout=1'b1**

3. Once the Kogge-Stone adder is implemented, create a testbench named **ksa_4bit_tb** and test your model with some representative values as inputs (at least 4 tests). Justify why you used these values in your submission. Submit screenshots with clear waveforms of your inputs and outputs and discuss your results (zoom and shift your waveform view in order to produce clear screenshots for your submission).
4. Submit a report, similar to previous labs, containing the information in the Deliverables section above.

Grading Breakdown:

- Design and implementation of red, yellow, and green blocks, along with final calculation of outputs – 50 pts
  - Module header and wire definitions – 10 pts
  - First row of P's and G's – 10 pts
  - Second row of P's and G's – 10 pts
  - Third row of P's and G's – 10 pts
  - Sum calculation – 10 pts

- Testbench implementation – 10 pts
- Use of separate files for model and testbench – 5 pts
- Justification for at least 4 input test vectors used – 10 pts
- Lab report with as much information present as possible, see Deliverables section – 20 pts
- Upload of .v model file and testbench (2 separate files) to Moodle and a .pdf report to Gradescope – 5 pts