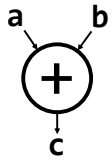
11 Dataflow Meets Logic [35 points]

We often use the "addition node":



to represent the addition of two input tokens. If we think of the tokens as binary numbers, we can model a simple logic circuit using dataflow graphs.[1] Note that a token can be used as an input to only *one* node. If the same value is needed by more than one node, it first should be replicated using one or more copy nodes, and then each copied token can be supplied to one node only.

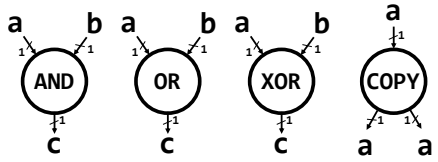
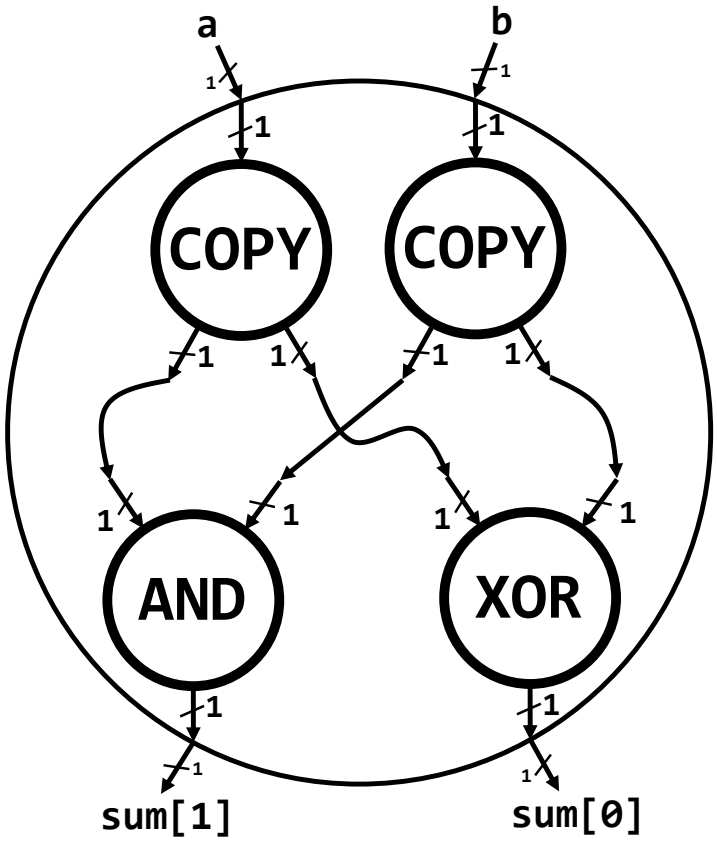


Figure 3: Dataflow nodes of basic bitwise operations allowed in Part (a).

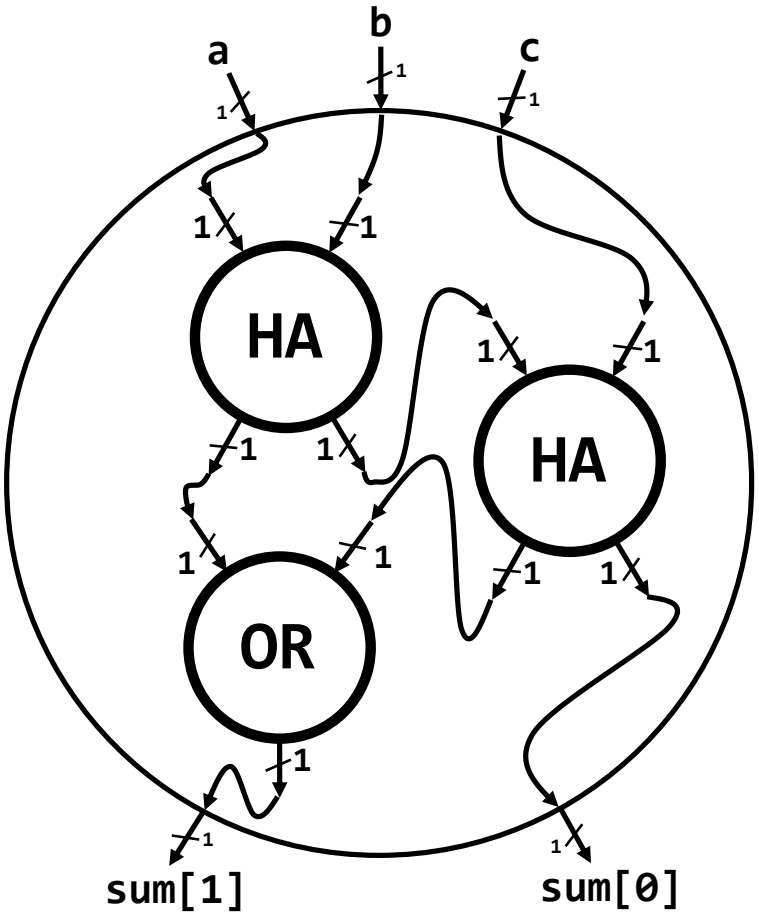
(a) [5 points] Implement the single-bit binary addition of two "1-bit" input tokens `a` and `b` as a dataflow graph using *only* 2-input {AND, OR, XOR} nodes and COPY nodes if necessary (illustrated in Figure 3). Fill in the internal implementation below, where inputs and outputs (labeled with their corresponding bit-widths) have been provided:



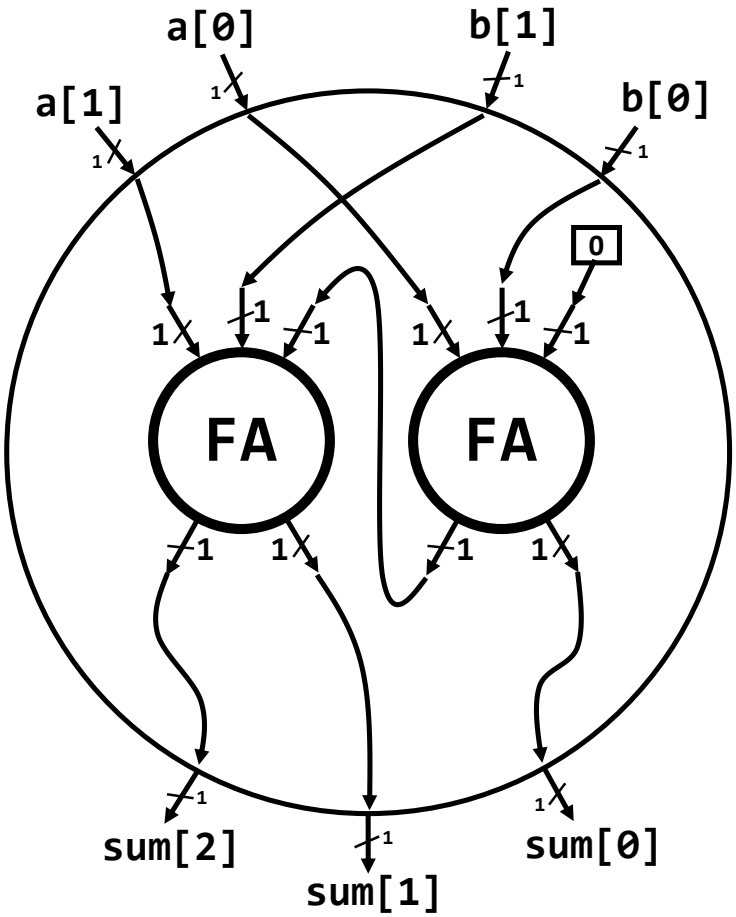
[1] Note: this is not an accurate electrical model of a circuit. Instead, the dataflow analogy is best thought of in terms of the desired flow of *information* rather than physical phenomena.

(b) [5 points] You may recognize the node we designed in part (a) as a model for a so-called "half-adder (HA)", which is not very useful by itself since it is only useful for adding 1-bit input tokens. In order to extend this design to perform binary addition of 2-bit input tokens `a[1:0]` and `b[1:0]`, the `sum[1]` token from half-adding `a[0]` and `b[0]` will have to act as an input token for *another* half-adder node used for adding `a[1]` and `b[1]`. This results in a 3-input adder called a "full-adder (FA)".

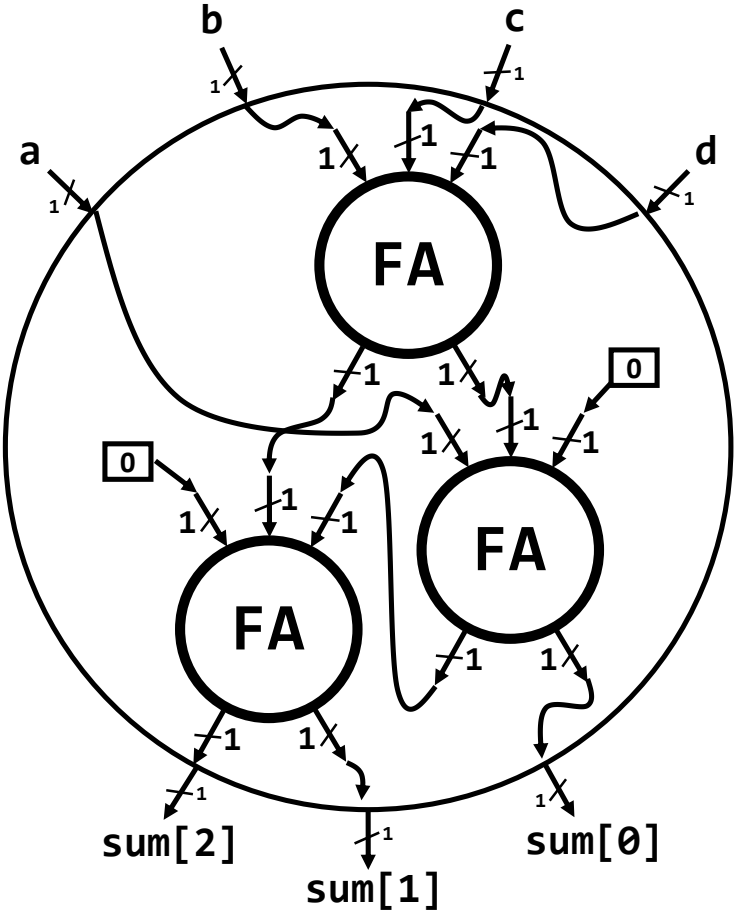
Fortunately, we can implement a full-adder (FA) using half-adders (HA) (i.e., the node we designed in part (a). Implement the full-adder using a *minimum* number of half-adders and *at most* 1 additional 2-input {AND, OR, XOR} node.



(c) [5 points] The full-adder (FA) is a versatile design that can be used to implement n-bit addition. Show how we might use it to implement 2-bit binary addition of two input tokens `a[1:0]` and `b[1:0]`. Use only a *minimum* number of full-adders (i.e., the dataflow node you designed in Part 2). *Hint: you may use constant input tokens if necessary.*



(d) [5 points] Interestingly, the full-adder can also be used to add four 1-bit input tokens. This is a natural extension of the full-adder in the same way we extended the half-adder to create the full-adder itself (in part (b)). Implement the 4-input node below using only a *minimum* number of full-adders (FA) (i.e., the dataflow node you designed in part (b)). *Hint: you may use constant input tokens if necessary.*



(e) [15 points] As it turns out, any $n \geq 3$ 1-bit input binary adders can be implemented purely using full-adders. Fill in the table below for the *minimum* number of required full adders to implement an n-input 1-bit adder.

n	# **required full-adders**
3	1
4	3
5	3
6	4
7	4
8	7