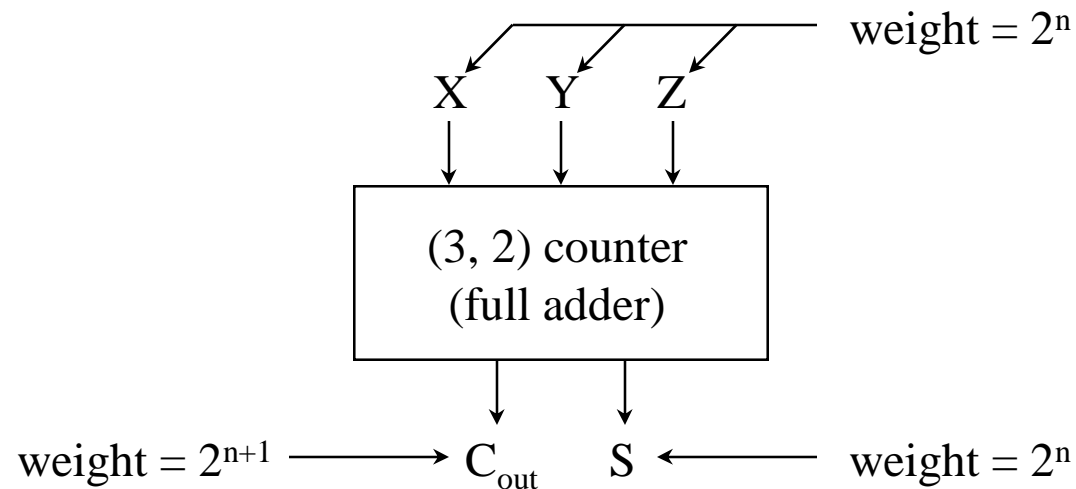


## Multi-Operand Adders

### (3, 2) Counter

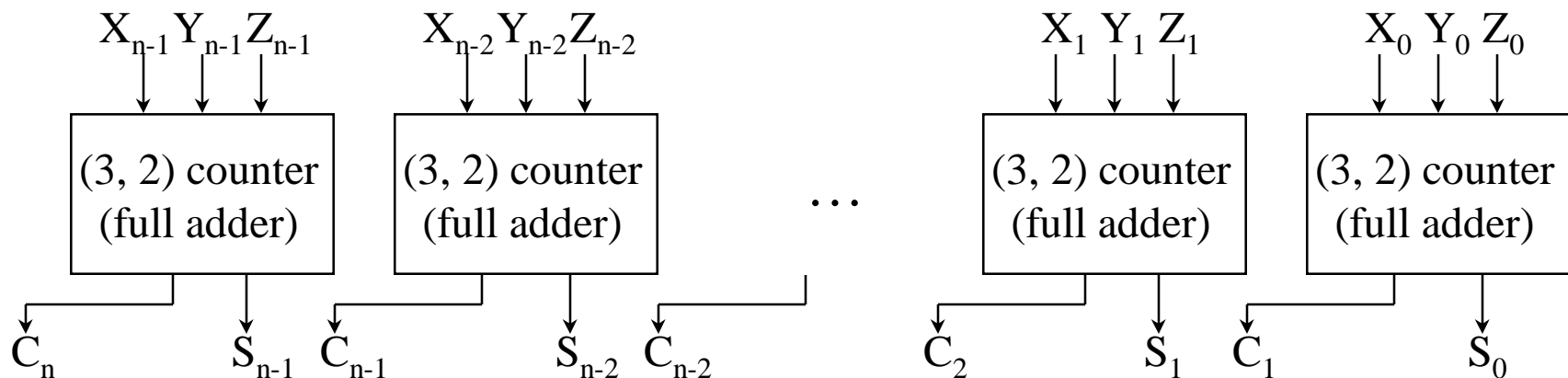
- An (m, n) counter takes as input m bits (all of the same power-of-2 weight) and produces an n-bit binary number whose value is the number of inputs that are equal to 1. In other words, it *counts* the number of 1s in the input and outputs the binary count value. Note that the outputs of the counter have *different* power-of-2 weights. The weight of the LSB of the counter output is the same as the weight of each of the inputs, and the remaining bits have increasingly higher weights.
- The simplest and most widely used example is the (3, 2) counter. Of the 3 inputs, there can be either 0, 1, 2 or 3 inputs equal to 1. All four of these values can be represented as a 2-bit binary number. In fact, the (3, 2) counter is nothing but a full adder, where the sum is the LSB count output and the carry-out is the MSB count output:



## Multi-Operand Adders

# Carry-Save Adder

- In a multiplier, we have to add many partial products together in order to obtain the final product. If we were to do this in a straightforward way, then we might successively accumulate partial products using a cascade of standard high-speed adders in which we have a carry propagation. We refer to such adders as carry propagate adders (CPAs). However, this would be very slow due to the carry propagation delay in each CPA.
- A much better alternative is to successively reduce 3 input vectors to 2 output vectors, i.e. a sum vector and a carry vector. In this way, each bit of these two vectors are computed independently of all other bits, and there is no carry propagation between adjacent bit positions. The hardware that implements a compression from 3 vectors X, Y and Z to 2 vectors S and C is called a carry-save adder (CSA). It is composed of a parallel set of (3, 2) counters, i.e. a parallel set of full adders.

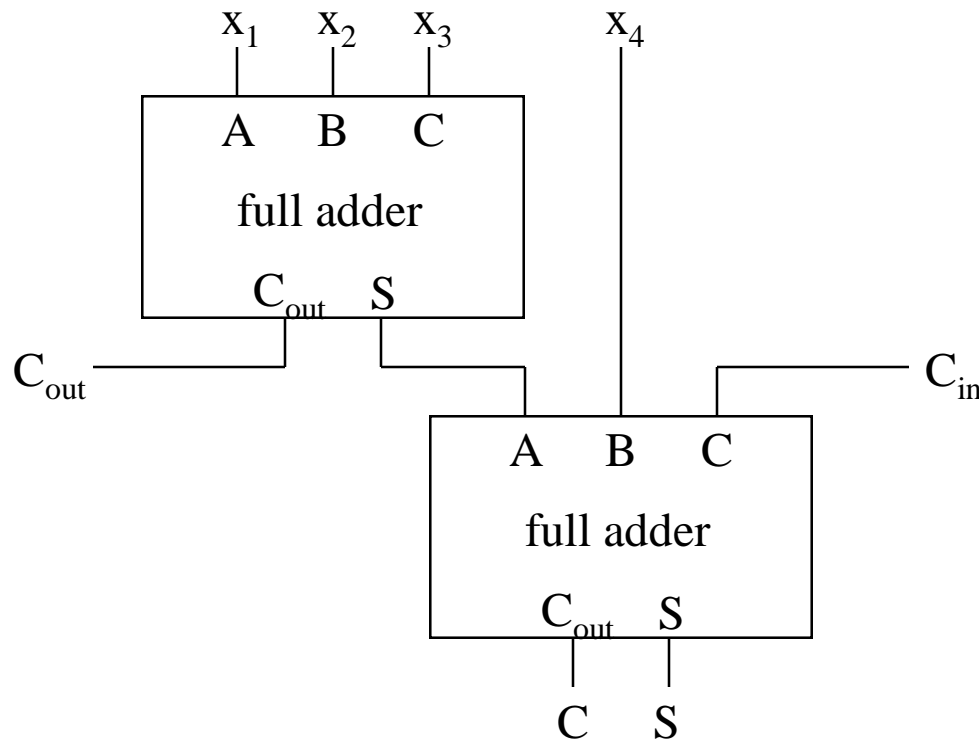




## Multi-Operand Adders

### [4, 2] Compressor: Basic Structure

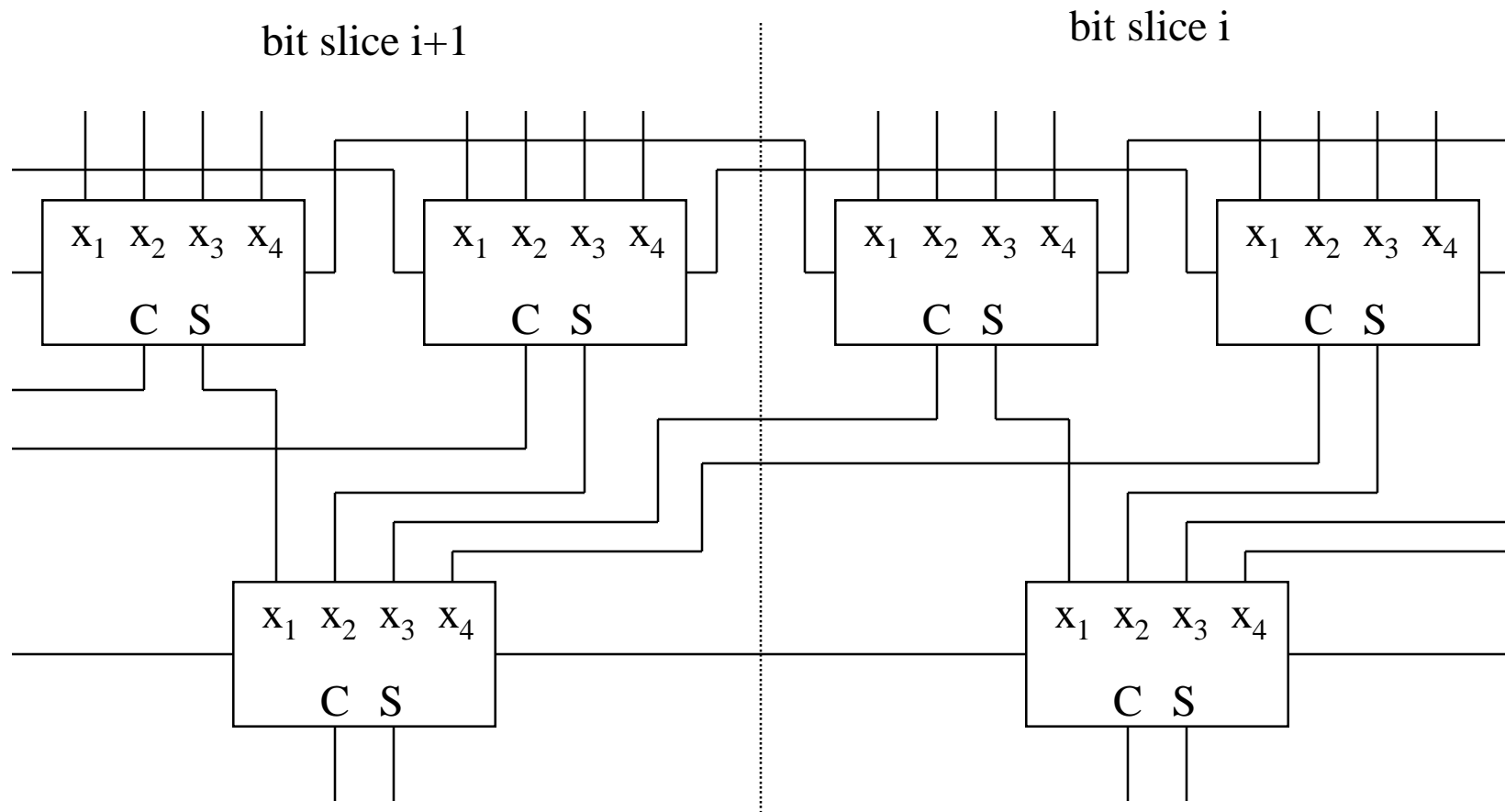
- Another useful element is the [4, 2] compressor. This unit actually has 5 inputs and 3 outputs. However, one of the inputs ( $C_{in}$ ) and one of the outputs ( $C_{out}$ ) are treated separately to minimize the total delay along any path. Aside from these, there are 4 inputs,  $x_1$ ,  $x_2$ ,  $x_3$ , and  $x_4$ , and 2 outputs,  $S$  and  $C$ . One possible implementation is to use two full adders. Each full adder could be built using 2 XORs and 3 MUXs, for a total of 4 XORs and 6 MUXs.



*Multi-Operand Adders*

## Tree Construction Using [4, 2] Compressors

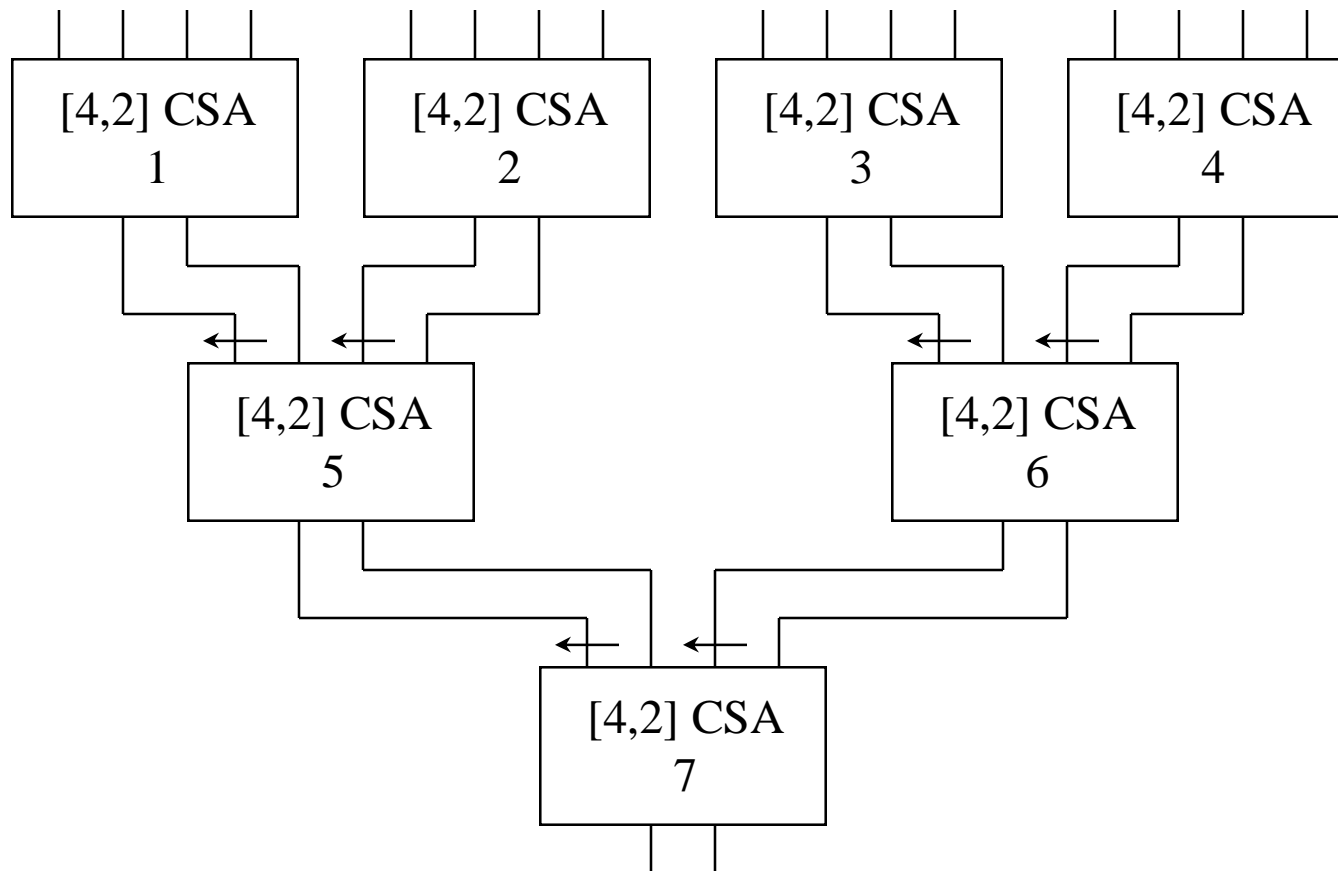
- Consider two adjacent bit slices in a tree that compresses 8 vectors to 2 vectors:



## Multi-Operand Adders

### [4, 2] Tree Construction

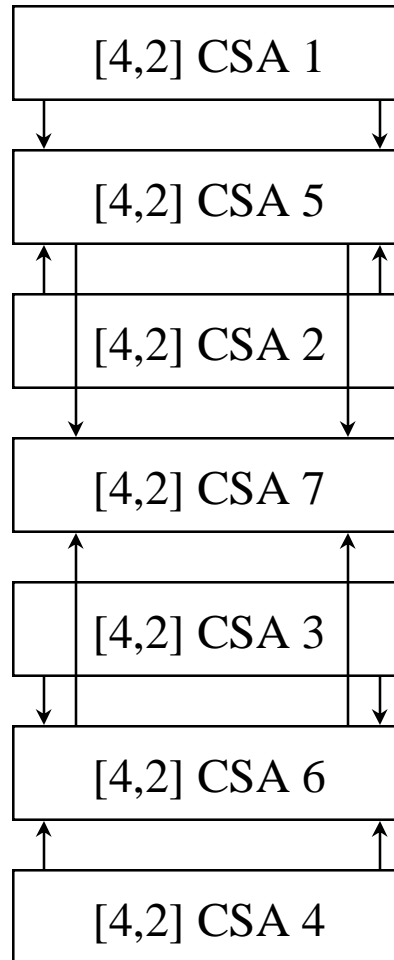
- We use the term “[4, 2] CSA” to mean a parallel set of [4, 2] compressors, including the appropriate connections to  $C_{in}$  and  $C_{out}$ . The binary tree structure leads to very regular layouts. An example of a tree for compressing 16 vectors to 2 vectors is shown below.



## *Multi-Operand Adders*

### [4, 2] Tree Layout

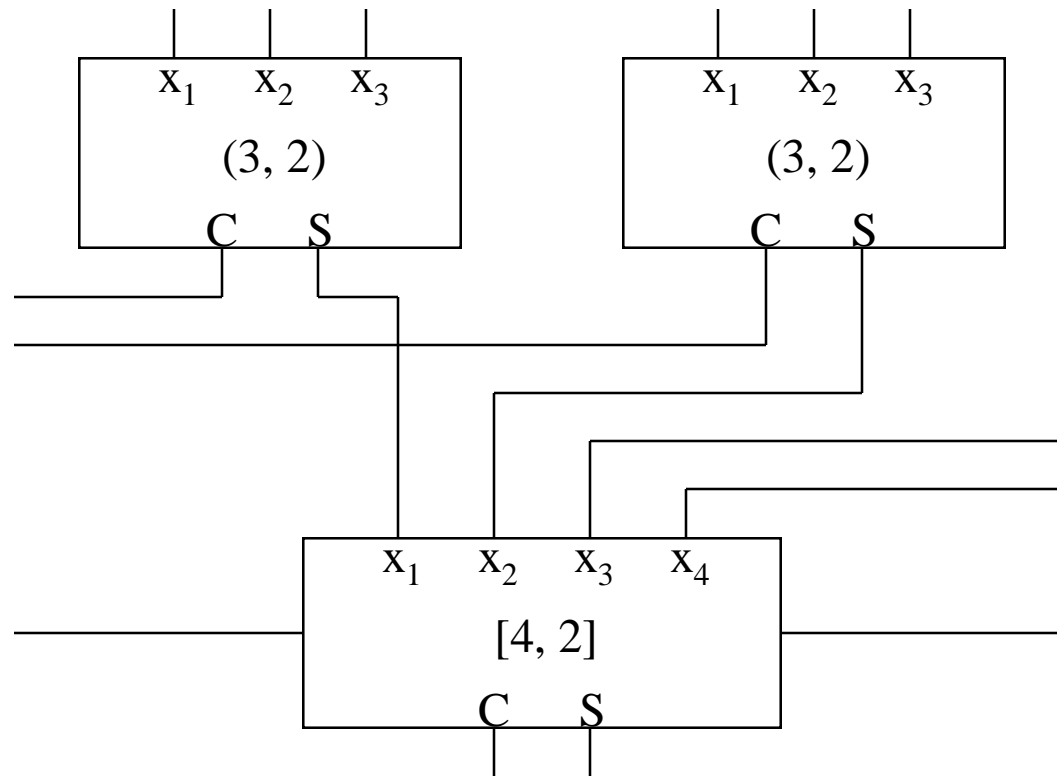
- The layout can be made even more regular by placing them in the order shown below:



*Multi-Operand Adders*

## [6, 2] Compressor

- Another possible tree structure can be built using [6, 2] compressors, each of which is built from two (3, 2) counters and one [4, 2] compressor, as shown below:



*Multi-Operand Adders*

# [9, 2] Compressor

- Yet another possible tree structure can be built using [9, 2] compressors, each of which is built from three (3, 2) counters and one [6, 2] compressor, as shown below:

