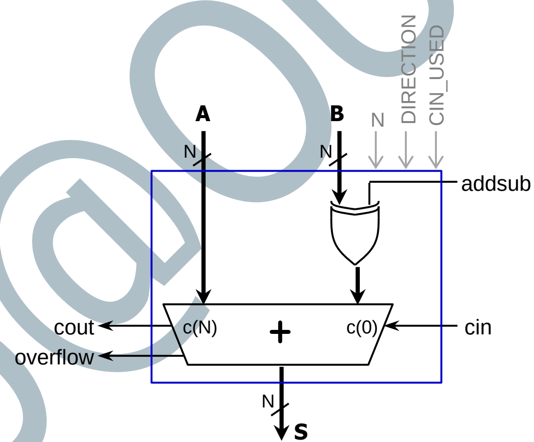
Adder/Subtractor with carry in

INTRODUCTION

- This is an important circuit in computer arithmetic. In particular the ability to incorporate a carry in (or borrow in) is a crucial requirement for specialized applications.
- A parameterized architecture is presented. Three parameters: DIRECTION, CIN_USED, and N. The circuit is described in VHDL using a purely structural approach based on full adders and logic gates.
  - ✓ The parameter N allows the selection of the size of the operation: N bits.
  - ✓ The parameter DIRECTION has 3 values: i) UNUSED: circuit includes an $addsub$ input for addition/subtraction selection, ii) ADD: circuit for only addition with carry in, and iii) SUB: circuit for only subtraction with an active-low borrow in.
  - ✓ The parameter CIN_USED has 2 values: i) YES: here, the carry in (cin) input is considered, and ii) NO: here, the carry in (cin) input is ignored; for addition, the default then is set 0, and for subtraction is 1.

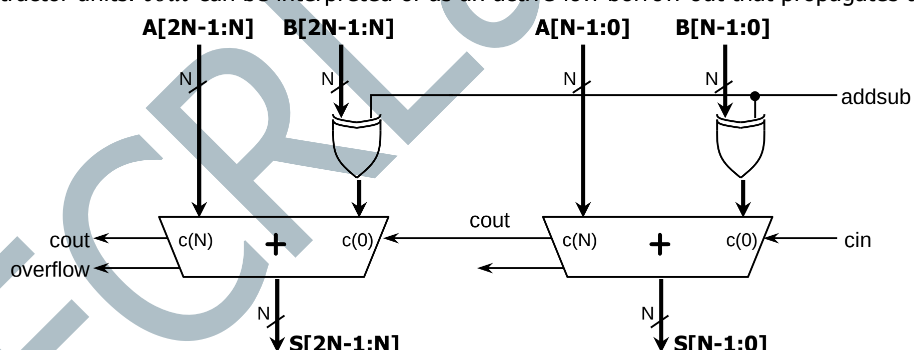
ADDER/SUBTRACTOR FOR SIGNED NUMBERS

- The table allows for the circuit in the figure. This is the standard adder/subtractor unit, where the cin input is an independent input.
- $cout = c(N)$, $overflow = c(N) \oplus c(N-1)$
- Addition: The operation is straightforward: $A + B + cin$
- Subtraction: We need to treat cin as an active-low borrow in. Thus, for signed numbers: $A - B = A + 2C(B) + cin - 1$.
  - ✓ If $cin = 0$, there is a borrow in and $A - B = A + 2C(B) - 1$.
  - ✓ If $cin = 1$, there is no borrow, and $A - B = A + 2C(B)$.



Operation	add_sub	cin	c(0)
ADDITION	0	0	0
	0	1	1
SUBTRACTION	1	0	0
	1	1	1

- The proposed approach works very well for multi-precision subtraction: this is when we partition the operation into two or more adder/subtractor units. $cout$ can be interpreted of as an active-low borrow out that propagates to the next unit.



- Note that if we were to treat cin as an active-high borrow in, $c(0)$ would depend on cin and $addsub$. Moreover, the circuit would not work well for multi-precision subtraction: the equation for $c(0)$ in the second (leftmost) subtractor would be different that for the first (rightmost) subtractor. The resulting circuit would become unnecessarily convoluted.

ADDER/SUBTRACTOR FOR UNSIGNED NUMBERS

- **ADDITION:** we use the exact same hardware (with carry in). $cout$ is the carry out bit and it also signals overflow. The overflow bit is only meaningful for signed operations.

- **SUBTRACTION:** We can use the subtractor for signed numbers. We need to zero extend the unsigned numbers to convert them to signed numbers. The operation is then a $(N+1)$ –bit addition. Also, $c(0) = cin$, which is an active-low borrow in.

$$\begin{array}{cccccc} 0 & A_{n-1} & A_{n-2} & \dots & A_0 & - \\ 0 & B_{n-1} & B_{n-2} & \dots & B_0 & \end{array} \Rightarrow$$

$A \geq B$:

$$\begin{array}{cccccc} c_n & c_{n-1} & c_{n-2} & & c_1 \; c_0 & \\ 0 & A_{n-1} & A_{n-2} & \dots & A_0 & + \\ 1 & /B_{n-1} & /B_{n-2} & \dots & /B_0 & \\ \hline 0 & S_{n-1} & S_{n-2} & \dots & S_0 & \end{array}$$

$A < B$:

$$\begin{array}{cccccc} c_n & c_{n-1} & c_{n-2} & & c_1 \; c_0 & \\ 0 & A_{n-1} & A_{n-2} & \dots & A_0 & + \\ 1 & /B_{n-1} & /B_{n-2} & \dots & /B_0 & \\ \hline 1 & S_{n-1} & S_{n-2} & \dots & S_0 & \end{array}$$

- ✓ If $A \geq B$, then $S_n = 0$. According to the figure, this only happens if $c(N) = 1$. The correct signed result is $0S_{n-1}S_{n-2} \dots S_0$. The correct unsigned result is $S_{n-1}S_{n-2} \dots S_0$.
- ✓ If $A < B$, then $S_n = 1$. According to the figure, this only happens if $c(N) = 0$. The correct unsigned result is $1S_{n-1}S_{n-2} \dots S_0$. The unsigned result is $S_{n-1}S_{n-2} \dots S_0$. This result is incomplete since a borrow out exists ($c(N) = 0$).

- $cout = c(N)$, and $cout$ can be interpreted as an active-low borrow out (as in the case for signed numbers). If $cout = 1$, then there is no borrow out. If $cout = 0$, there is a borrow out.

- Since we are only considering $S_{n-1}S_{n-2} \dots S_0$ and $c(N)$, we notice that we do not need to actually perform zero-extension in the circuit: we just use the same adder/subtractor circuit and it is up to the user to treat the inputs as signed or unsigned. If the inputs are treated as unsigned, the overflow output is meaningless.