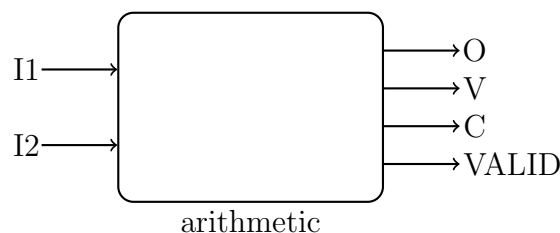


# Arithmetic

Ihre Aufgabe ist es, das Verhalten einer Entity namens „arithmetic“ zu programmieren. Die Entity ist in der angehängten Datei „arithmetic.vhdl“ deklariert und hat folgende Eigenschaften:

- Eingänge: I1, I2 vom Typ `std_logic_vector`
- Ausgänge: O vom Typ `std_logic_vector`, V vom Typ `std_logic`, C vom Typ `std_logic`, VALID vom Typ `std_logic`



Verändern sie die Datei „arithmetic.vhdl“ nicht!

Die Entity soll folgende arithmetische Funktionen implementieren:

- Addition  $I1 + I2$
- Eingangsoperand (I1): 15 Bit, Zweierkomplement
- Eingangsoperand 2 (I2): 12 Bit, Zweierkomplement
- Ausgang (O): 15 Bit, Zweierkomplement
- Overflow (V) und Carry Flag (C) entsprechend gesetzt
- Valid Flag (VALID): Gibt an, ob die berechnete Lösung gültig ist oder nicht.

Dieses Verhalten soll von Ihnen in der Datei „arithmetic\_beh.vhdl“ programmiert werden.

**Anmerkung:** Die ALU soll mit einer Bitlänge von 15 implementiert werden. Daher kann eine Vorzeichenerweiterung der Operanden notwendig sein.

## Zusammenfassung: Carry und Overflow Flag

### Carry Flag

Das Carry Flag wird in der binären Mathematik unter zwei Bedingungen gesetzt ( $= '1'$ ):

1. Das Carry Flag wird gesetzt, wenn bei der Addition von zwei Zahlen das MSB (höchstwertigste Bit) des Ergebnisses aus den verfügbaren Bit-Stellen „herausgescho-ben“ wird.

z.B.  $1111 + 0001 = 0000$  (Carry Flag ist '1')

2. Das Carry (Borrow) Flag wird ebenfalls bei der Subtraktion zweier Zahlen gesetzt, wenn dabei an der Stelle des MSB ein Bit „ausgeborgt“ werden muss.

z.B.  $0000 - 0001 = 1111$  (Carry Flag ist '1')

Ansonsten ist das Carry Flag immer nicht gesetzt ( $= '0'$ ).

- z.B.  $0111 + 0001 = 1000$  (Carry Flag ist '0')
- z.B.  $1000 - 0001 = 0111$  (Carry Flag ist '0')

In vorzeichenloser Arithmetik zeigt ein gesetztes Carry Flag, dass das Ergebnis falsch ist.

In vorzeichenbehafteter Arithmetik liefert das Carry Flag keine relevanten Informationen.

### Overflow Flag

Das Overflow Flag wird in der binären Mathematik unter folgenden Bedingungen gesetzt ( $= '1'$ ):

1. Wenn die Summe zweier Zahlen, welche jeweils kein gesetztes Vorzeichenbit aufweisen, zu einem Ergebnis mit einem gesetzten Vorzeichenbit führt:

z.B.  $0100 + 0100 = 1000$  (Overflow Flag ist '1')

2. Wenn die Summe zweier Zahlen, welche jeweils ein gesetztes Vorzeichenbit aufweisen, zu einem Ergebnis mit einem nicht gesetzten Vorzeichenbit führt:

z.B.  $1000 + 1000 = 0000$  (Overflow Flag ist '1')

3. Wenn die Subtraktion einer Zahl mit gesetztem Vorzeichenbit von einer Zahl mit nicht gesetztem Vorzeichenbit zu einem Ergebnis mit gesetztem Vorzeichenbit führt:

z.B.  $0111 - 1001 = 1110$  (Overflow Flag ist '1')

4. Wenn die Subtraktion einer Zahl mit nicht gesetztem Vorzeichenbit von einer Zahl mit gesetztem Vorzeichenbit zu einem Ergebnis mit nicht gesetztem Vorzeichenbit führt:

z.B.  $1001 - 0111 = 0010$  (Overflow Flag ist '1')

Ansonsten ist das Overflow Flag immer nicht gesetzt ( $= '0'$ ).

- z.B.  $0100 + 0001 = 0101$  (Overflow Flag ist '0')
- z.B.  $0110 + 1001 = 1111$  (Overflow Flag ist '0')
- z.B.  $1000 + 0001 = 1001$  (Overflow Flag ist '0')
- z.B.  $1100 + 1100 = 1000$  (Overflow Flag ist '0')

Bei vorzeichenbehafteter Arithmetik zeigt ein gesetztes Overflow Flag, dass das Ergebnis falsch ist.

Bei der vorzeichenlosen Arithmetik liefert das Overflow Flag keine relevanten Informationen.

---

Tipp: Verwenden Sie das „ieee.numeric\_std“ Package für die Implementierung der arithmetischen Operation.

Um Ihre Lösung abzugeben, senden Sie ein E-Mail mit dem Betreff „Result Task 8“ und Ihrer Datei „arithmetic\_beh.vhdl“ an [vhdl-mc+e384@tuwien.ac.at](mailto:vhdl-mc+e384@tuwien.ac.at).

Viel Erfolg und möge die Macht mit Ihnen sein.