Σύνθεση υψηλού επιπέδου για τη σχεδίαση ψηφιακών ολοκληρωμένων κυκλωμάτων

2ο σετ ασκήσεων

Άσκηση 1η

(α) Για το παράδειγμα που σας δίνεται επιλέξτε το ελάχιστο πλήθος από μπιτς που απαιτείται ώστε να αναπαρασταθεί πλήρως το αποτέλεσμα.

```
ac_int<8,false> a, b; // 8-bit unsigned integer
ac_int<9,true> c, d; // 9-bit signed integer
ac_int<?,true> e = a*b + c*d; // How many bits
```

(β) Για το παράδειγμα που σας δίνεται επιλέξτε το ελάχιστο πλήθος από μπιτς που απαιτείται ώστε αναπαρασταθεί με πλήρως και με ακρίβεια το αποτέλεσμα.

```
ac_fixed<5,2,false> f; //5-bit unsigned fixed-point type with 2 integer bits
ac_fixed<5,4,true> g; // 5-bit signed fixed-point type with 4 integer bits
ac_fixed<?,?,true> h = f*g; // How many bits
```

Ελέγξτε το αποτελέσμα για το πλήθος των μπιτς που διαλέξατε με κατάλληλα πειράματα σε C++.

Άσκηση 2η

Καλείστε να γράψετε δύο συναρτήσεις σε C++. Η πρώτη συνάρτηση δέχεται στην είσοδο της ένα προσημασμένο ακέραιο των W μπιτς (ac_int<W,true>) και υπολογίζει την canonic signed digit (CSD) αναπαράσταση του. Η δεύτερη συνάρτηση δέχεται την αναπαράσταση CSD μιας σταθερας και έναν προσημασμένο ακέραιο και υπολογίζει το γινόμενο τους με ολισθήσεις και προσθαφαιρέσεις.

Η αναπαράσταση CSD αναπαριστά τους προσημασμένους ακέραιους με τα ψηφία $\bar{1}, 0, 1$ εξασφαλίζοντας ταυτόχρονα τη χρήση των ελάχιστων μη-μηδενικών ψηφίων. Για παράδειγμα ο αριθμός $011101 = 29$ μπορεί να αναπαρασταθεί σαν $100\bar{1}01 = 32 - 4 + 1$ μειώνοντας έτσι τα μη-μηδενικά στοιχεία απο 4 σε 3.

Στον παρακάτω σύνδεσμο μπορείτε να βρείτε μια αναλυτική περιγραφή της αναπαράστασης CSD και ενός αλγορίθμου μετατροπής προσημασμένων ακεραίων σε CSD.

https://www.allaboutcircuits.com/technical-articles/an-introduction-to-canonical-signed-digit-representation/

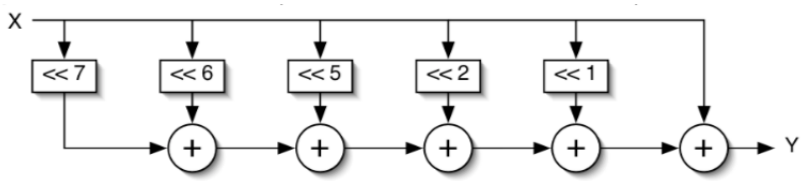
Η αποθήκευση κάθε αριθμού CSD απαιτεί δυο σειρές από μπιτ x^+ και x^-. Τα ψηφία του x^+ αναπαριστούν τα θετικά και τα μηδενικά ψηφία του αριθμού ενώ τα x^- αναπαριστούν τα αρνητικά. Έτσι ο αριθμός $100\bar{1}01$ που ακολουθεί την αναπαράσταση CSD αποθηκεύεται ως $x^+ = 100001$ και $x^- = 000100$. Η τιμή ενός τέτοιου αριθμού είναι πάντα ίση με $x^+ - x^-$. Με βάση αυτό η συνάρτηση σας θα μπορούσε να έχει τον εξής ορισμό:

```
template <int W>
void csd_encode (ac_int<W,true> &num, ac_int<W,false> &x_p, ac_int<W,false>
&x_m){
  // num is the input number to be encoded in CSD
  // x_p and x_m are the two bit vectors that represent the CSD number
}
```

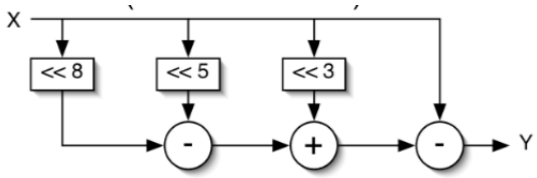
Εναλλακτικά θα μπορείτε να ορίσετε μία δομή για τους αριθμούς CSD και να χρησιμοποιήσετε αυτήν στα ορίσματα της συνάρτησης σας.

```
template <int W>
struct CSD {
  ac_int<W,false> x_p;
  ac_int<W,false> x_m;
};
...
template <int W>
void csd_encode (ac_int<W,true> &num, CSD<W> &num_csd)
```

Αφού μπορείτε πλέον να υπολογίζετε την αναπαράσταση CSD ενός αριθμού ή μιας σταθεράς μπορείτε να υλοποιήσετε πιο φθηνά πράξεις πολλαπλασιασμού ενός ακέραιου με μία σταθερά χρησιμοποιώντας διαδοχικές ολισθήσεις και προσθαφαιρέσεις. Για παράδειγμα η πράξη $231 * X$ θα μπορούσε να υπολογιστεί ως $(2^7 + 2^6 + 2^5 + 2^2 + 2^1 + 2^0) * X$



ή φθηνότερα, με λιγότερες ολισθήσεις και προσθαφαιρέσεις, ως $(2^8 - 2^5 + 2^3 - 2^0) * X$ το οποίο προκύπτει από την αναπάρασταση σε CSD $100\bar{1}0100\bar{1}$ της σταθεράς 231.



Η συνάρτηση του φθηνού πολλαπλασιασμού ενός αριθμού με μία σταθερά θα μπορούσε να οριστεί ως εξής:

```
ac_int<Wout,true> csd_mult (ac_int<W,true> &in, const CSD<W> &constant_csd)
  // in is the input value (non constant)
  // constant_csd is the CSD representation of a constant
}
```

Για τις ολισθήσεις και τις προσθαφαραίσεις πρέπει να χρησιμοποιήσετε έτοιμους τελεστές της κλάσσης ac_int.

Σε κάθε περίπτωση μην ξεχάσετε να δημιουργήσετε ένα πρόγραμμα ελέγχου σε C++ που θα τροφοδοτεί τις συναρτήσεις σας με ψευδοτυχαίες εισόδους.

Άσκηση 3η

Θέλετε να σχεδιάσετε ένα κύκλωμα κωδικοποίησης run length. Το κύκλωμα σας δέχετε μέσω ενός καναλιου ac_channel τετραμπιτους ακεραίους (δλδ ac_channel<ac_int<4,false> > &in). Αφού σιγουρέψετε ότι το κανάλι σας έχει τουλάχιστον 10 τιμές αποθηκευμένες πρέπει να τις κωδικοποιήσετε ώστε συνεχόμενες εκδοχές της ίδιας τιμής να μεταφέρονται ως η τιμή και πόσες φορές εμφανίστηκε. Για παράδειγμα η 10αδα, 2 2 2 2 3 12 12 1 1 5 θα μπορούσε να κωδικοποιηθεί ως 2 4 3 1 12 2 1 2 5 1 (4 φορες το 2, 1 φορά το 3, 2 φορές το 12, 2 φορές το 1 και 1 φορά το 5). Η κωδικοποίηση από διαδοχικά ζεύγη (τιμή, εμφανίσεις) θα εμφανίζεται σειριακά σε ένα κανάλι εξόδου ac_channel<ac_int<4,false> > &out κάθε φορά που είναι έτοιμη. Παράλληλα με το κύκλωμα σας φροντίσετε να δημιουργήσετε ένα ευέλικτο πρόγραμμα ελέγχου σε C++.

```
void runlength_encode(ac_channel<ac_int<4,false> > &in,
                      ac_channel<ac_int<4,false> > &out)
```