# Σύνθεση υψηλού επιπέδου για τη σχεδίαση ψηφιακών ολοκληρωμένων κυκλωμάτων

3ο σετ ασκήσεων

## 'Ασκηση 1η

Επαναλάβετε την 3η άσκηση του 2ο σετ ασκήσεων που αφορούσε στη σχεδίαση του κωδικοποιητή run_length. Αυτή τη φορά **η μόνη εγγύηση που αναζητάτε** είναι ότι **το κανάλι έχει το πολύ μία νέα τιμή διαθέσιμη**. Επομένως θα πρέπει το κύκλωμα σας να θυμάται τον τελευταίο σύμβολο που πέρασε καθώς και πόσες φορές το είχε δεχθεί μέχρι τώρα. Αποφύγετε τις στατικές και τις καθολικές μεταβλητές και αποθηκεύστε την πληροφορία που χρειάζεστε σαν ιδιωτικές μεταβλητές μία κλάσης σύμφωνα με το παρακάτω πρότυπο.

```
typedef ac_int<4,false> dtype;

class RunLengthEncoder {
private:
  // internal state
public:
   // constructor - init internal state
  RunLengthEncoder() {...}
  // top-level interface
  void run (ac_channel<dtype> &in, ac_channel<dtype> &out) {...}
};
```

Μην ξεχάσετε να προσαρμόσετε κατάλληλα το πρόγραμμα ελέγχου για το νέο κύκλωμα.

## 'Ασκηση 2η

Σας ζητείτε να σχεδιάσετε μία μονάδα υλικού σε C++ που να ελέγχει το αποτέλεσμα του παιχνιδιού blackjack και στη συνέχεια θα περάσει από σύνθεση υψηλού επιπέδου (θα το κάνουμε μαζί στο 2ο εργαστήριο) . Οι νέες κάρτες (τετράμπιτοι θετικοί ακέραιοι) καταφθάνουν σε κάθε γύρο του παιχνιδιού μέσω ενός καναλιού. Ο ελεγκτής αθροίζει την τιμή της κάθε κάρτας και υπολογίζει το τρέχον άθροισμα. Οι κάρτες 2-10 μετρούν όσο το όνομα τους, ενώ, η κάρτα 1 που αναπαριστά τον άσσο μετρά για 11. Μεγαλύτερες κάρτες δε μπορούν να εμφανιστούν. Ο παίκτης κερδίζει αν οι κάρτες που τράβηξε έχουν άθροισμα ίσο με 21, ενω καίγεται αν είναι μεγαλύτερο του 21. Η μόνη περίπτωση που κερδίζει παρότι το άθροισμα υπερβαίνει το 21 είναι να έχει δύο άσσους στα δύο πρώτα φύλλα που τράβηξε. Όταν ο παίκτης κερδίσει ή καεί σταμάταει αυτόματα ο τρέχων γύρος. Ο παίκτης που δεν έχει κερδίσει ή δεν έχει καεί μπορεί να συνεχίσει να τραβάει μια νέα κάρτα. Αν έχει τραβήξει 5 κάρτες που ακόμη έχουν άθροισμα μικρότερο του 21 τότε και πάλι θεωρείται νικητής. Ο κάθε γύρος δε μπορεί να

περιλαμβάνει περισσότερα από 5 φύλλα.

Ο ελεγκτής ενημερώνει για την κατάσταση του παιχνιδιού μέσω δυο μεταβλητών end_round και win. Με end_round = true ενημερώνει ότι ο τρέχων γύρος τελείωσε (ή γιατί ο παίκτης κερδισε ή γιατί κάηκε). Την ίδια στιγμή η τιμή του win δείχνει αν ο γύρος τελείωσε με νίκη ή ήττα. Υλοποιήστε τον ελεγκτή του blackjack μέσω μίας κλάσης που ακολουθεί το παρακάτω πρότυπο:

```
typedef ac_int<4,false> Card;

class BlackJack {
private:
  // internal state
public:
  // constructor - init internal state
  BlackJack() { ... }
  // top-level interface
  void run (ac_channel<Card> &in_card,
            bool &end_round,
            bool &win) {...}
};
```

Όπως και στην 1η άσκηση αυτού του σετ από το κανάλι εισόδου θα απορροφάτε μία μία τις κάρτες χωρίς να περιμένετε να είναι διαθέσιμες και οι πέντε κάρτες που θα κάλυπταν ένα γύρο.

Στη συνάρτηση main του προγράμματος σας σε C++ πρέπει να συμπεριλάβετε ένα πλήρες testbench που θα στέλνει ψευδοτυχαίες τιμές καρτών και θα παρατηρεί την έξοδο του ελεγκτή. Το testbench θα χρειαστεί οπωσδήποτε καθώς στο 2ο εργαστήριο θα επαναλάβουμε πως μπορείτε να κάνετε προσομοίωση της παραχθείσας RTL στο Questasim (σαν το Modelsim) επαναχρησιμοποιώντας με αυτόματο τρόπο το testbench σας σε C++.