**Exemple:**

```
a(bc)*   - se va potrivi cu şirurile de caractere 'a', 'abc', 'abcbc', 'abcbcbc', etc.
(ab)+c   - se va potrivi cu şirurile de caractere 'abc', 'ababc', 'abababc', etc.
ab+a     - se va potrivi cu şirurile de caractere 'aba', 'abba', 'abbba', etc.
```

Nu confundați operatorii "*" și "+" cu înmulțire și adunare. În contextul expresiilor regulate, aceștia sunt folosiți pentru a descrie tipare de căutare și **NU** sunt folosiți pentru a scrie cod Verilog.

**Debouncing**

Atunci când un buton este apăsat sau un switch este comutat, două părţi metalice intră în contact pentru a permite curentului să treacă. Cu toate acestea, ele nu se conectează instantaneu, ci se conectează și deconectează de câteva ori înainte de realizarea conexiunii propriu-zise. Acelaşi lucru se întâmplă și în momentul eliberării unui buton (când acesta nu mai este apăsat). Acest fenomen poate conduce la comutări false sau modificări multiple nedorite asupra semnalului și este denumit **bouncing**.

Prin urmare, se poate spune că fenomenul de "bouncing" nu este un comportament ideal pentru niciun switch care execută mai multe tranziţii ale unei singure intrări. Aceasta nu este o problemă majoră când avem de-a face cu circuite de putere, dar poate cauza probleme atunci când avem de-a face cu circuitele logice sau digitale. Aşadar, pentru a elimina oscilaţiile din semnal cauzate de acest fenomen se foloseşte principiul de **Switch Debouncing**.

Procedeul de debouncing este întâlnit, de asemenea și în software. Urmăriţi în scheletul de cod din exerciţiul 1 cum este implementat un debouncer și analizaţi comportamentul acestuia.

**Exerciţii**

1. Se doreşte proiectarea unui automat finit capabil să recunoască secvenţe de tip "ba". Automatul primeşte la intrare în mod continuu caractere codificate printr-un semnal de un bit (caracterele posibile sunt “a” și “b”). Ieşirea automatului va consta dintr-un semnal care va fi activat (valoarea 1) atunci când la intrare am avut prezent un șir care se potriveşte cu tiparul de căutare.

a. Implementaţi automatul în Verilog.

- *Hint*: Realizaţi pe hârtie schema automatului de stări, pentru a o folosi ulterior ca referinţă.
- *Hint*: Observaţi în laboratorul 0 strategia abordată pentru implementarea unui automat ce recunoaşte o secvenţă de caractere.

b. (**2p**) Simulaţi automatul folosind modulul de test din scheletul de cod. Eliminaţi semnalele nerelevante (*is* și *count*) din diagrama de semnale. Adăugaţi starea automatului și starea următoare a automatului la diagrama de semnale.

- *Hint*: Semnalele pot fi eliminate din diagrama de semnale cu *click dreapta->Delete* pe semnalul care se doreşte a fi eliminat.

- *Hint*: Semnale noi pot fi adăugate la diagrama de semnale prin *drag-and-drop* din fereastra *Simulation Objects for ...*, care conţine toate semnalele modulului selectat în fereastra *Instance and Process Name*.
- *Hint*: Simularea trebuie repornită prin *Simulation->Restart* urmat de *Simulation->Run* pentru a vedea comportamentul semnalelor adăugate.

c. Urmăriţi diagrama de semnale şi codul automatului şi explicaţi comportamentul. Urmăriţi şi explicaţi funcţionarea modulului de test.

2. Se dorește realizarea unei treceri de pietoni semaforizate. Duratele de timp pentru cele 2 culori vor fi: roşu - 60 sec, verde - 30 sec.

a. Implementaţi şi simulaţi în Verilog automatul necesar. Ce rol are modulul *trecere* din fişierul *trecere.v*?

- *Hint*: Consultaţi laboratorul 0 pentru diagrama de tranziţie a unui astfel de automat.

b. Explicaţi codul numărătorului din fişierul *counter.v*.

- *Hint*: Urmăriţi comportarea acestuia pe diagrama de semnale.