RĪGAS TEHNISKĀ UNIVERSITTĀTE

ELEKTRONIKAS UN TELEKOMUNIKĀCIJU FAKULTĀTE

[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwj9zKXEs6fgAhWF1iwKHceJAlAQjRx6BAgBEAU&url=https://lv.wikipedia.org/wiki/Att%C4%93ls:RTU_logo_2017.svg&psig=AOvVaw2YftzNrZHl3gHZWIEXy3Ih&ust=1549552627423236)

Funkcionālo un loģisko shēmu modelēšana, RTR532

Laboratorijas darbs Nr.5

Bogdans Žukovskis

161REC020

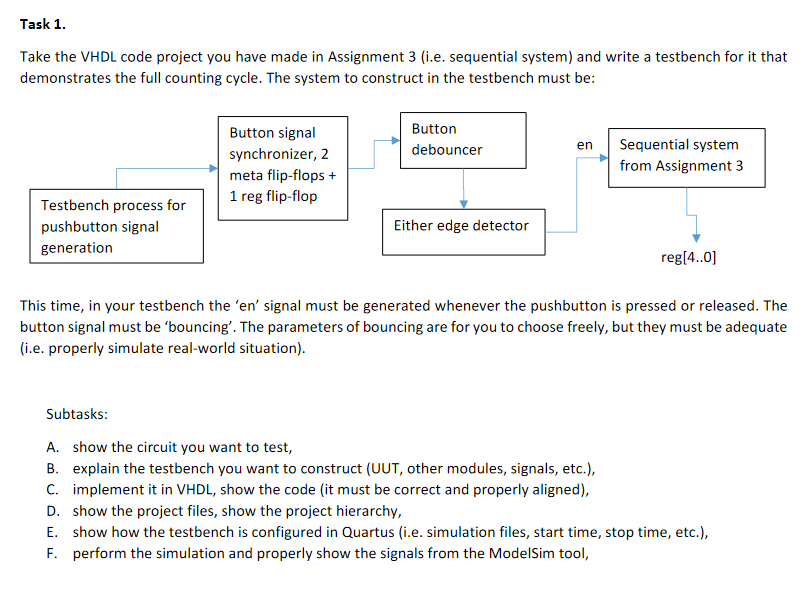
Riga, 2021

# Ievads

Variants:

5. laboratorijas darbs ir veltīts skaitītāja no 3. uzdevuma testēšanai, imitējot asinhrono pogas nospiešanu, kura kontrole enable signālu. Tāda veida radot nepieciešamību papildināt skaitītāja shēmu ar sinhronizatoru un malu detektēšanas ķēdi (edge detection circuit).

# Uzdevums



## B.

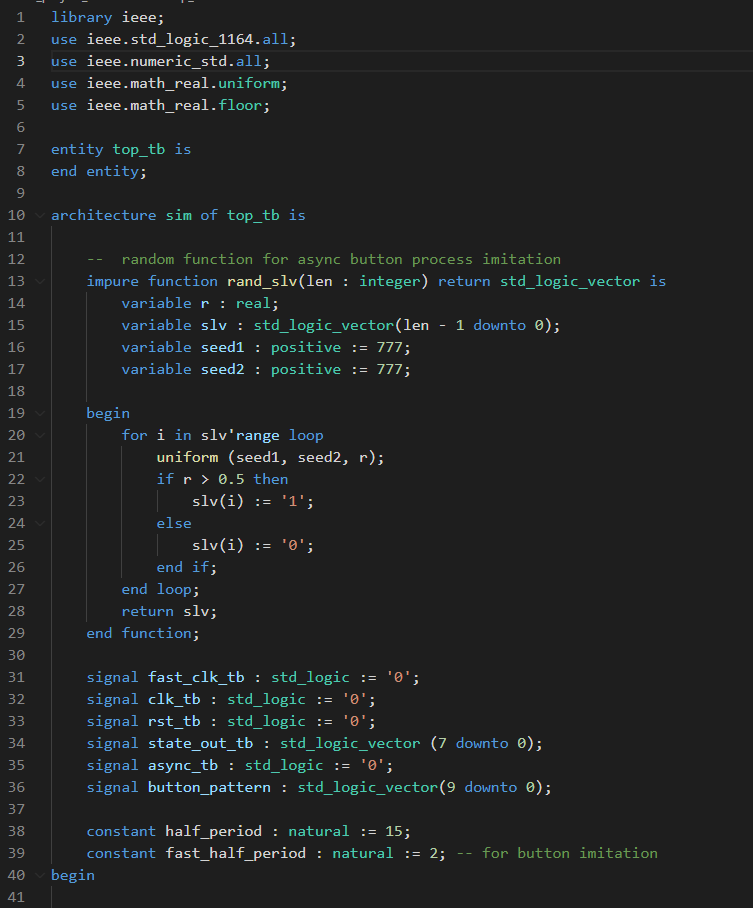
Klasiska testbench struktūra ietver sevī ieejas ģenerēšanu, moduli kas tiek testēts un izejas rezultātu salīdzināšana ar sagaidāmo. Mūsu gadījumā tiek testēta secīga shēma , kura uzvadās kā skaitītais ar dažiem ekstra stāvokļiem , kur stāvokļu parēja nav regulāra (+1). *Enable* signāla ģenerēšanai būs izmantots asinhrons process testbench iekšā, tāpēc skaitītais ir papildināts ar sinhronizatoru un malu noteikšanas ķēdi.

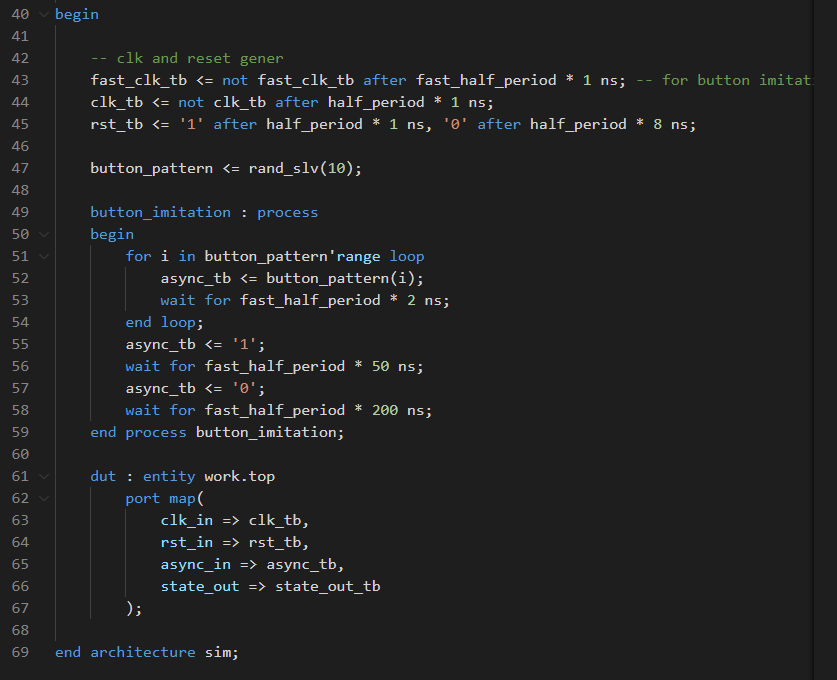
Testbench būs vienkāršs – funkcionāla manuāla simulācija. Tas nozīme ka izejas tiek pārbaudītas bez automātikas un signālu pārvadīšanā nav iesaistītas aiztures. Testējamas shēmas stāvokļa reģistram ir *enable* ieeja, kura kontrolēsies no asinhrona procesa, kurš imitēs pogas nospiešanu. Top modulis uzvedības veidā aprakstā skaitītāju un strukturāli ietver sevi sinhronizatoru ar malu detektēšanas ķēdi. *Top* moduļa saskārne ietvēr sevī *taktēšanas* signālu, *reset* , *asinhrono* *ieeju* un *stāvokļa* *izeju*. Top testbench failā asinhrona ieeja ir ģenerēta ar pseudo random funkcijas palīdzību. Lai process būtu asinhrons bija saģenerēts ātrs taktēšanas signāls, kura periods nav vairākkārtējs galvenajam taktēšanas signālam. Signāli kuri ir definēti testbench iekšā ir ar *\_tb* pierakstu. Ir vērts pamanīt, ka rakstot simulācijas failu var arī parastajiem signāliem uzdot sākuma vērtību. Ko labāk nedarīt sintezējamā koda gadījumā. Sinhronizators ar mālu detektēšanas ķēdi ir aprakstīti *synchr\_edge\_det.vhd* faila iekšā *work* library iekšā.

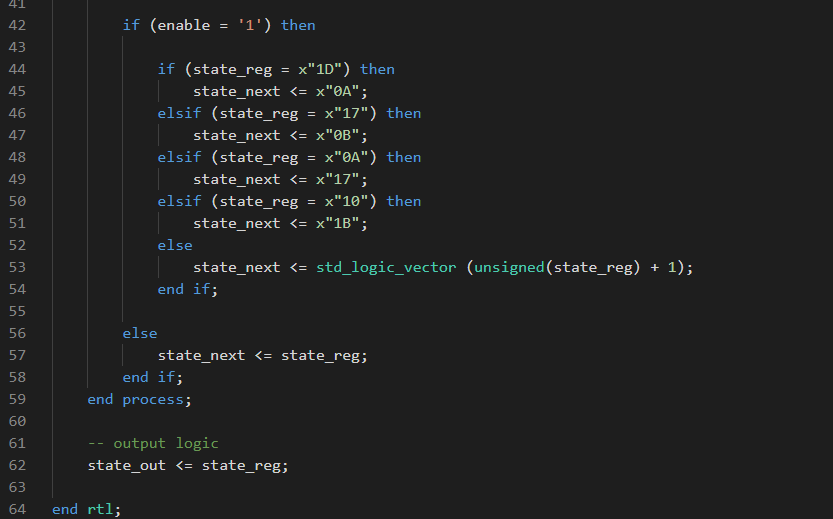
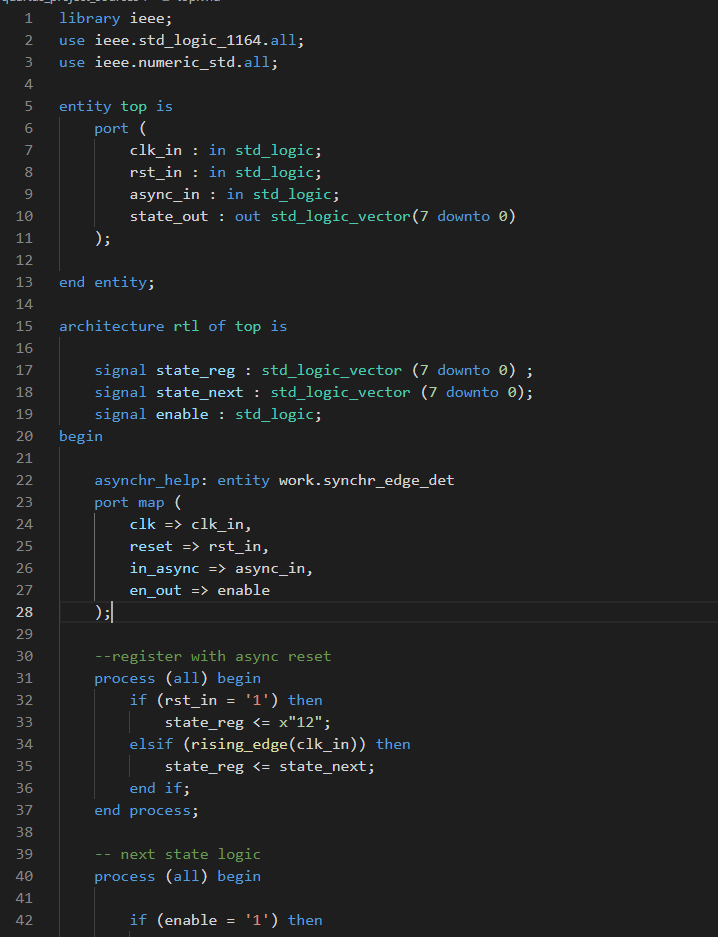
## C.

Visiem aprakstītiem failiem kods izskatās sekojoši:

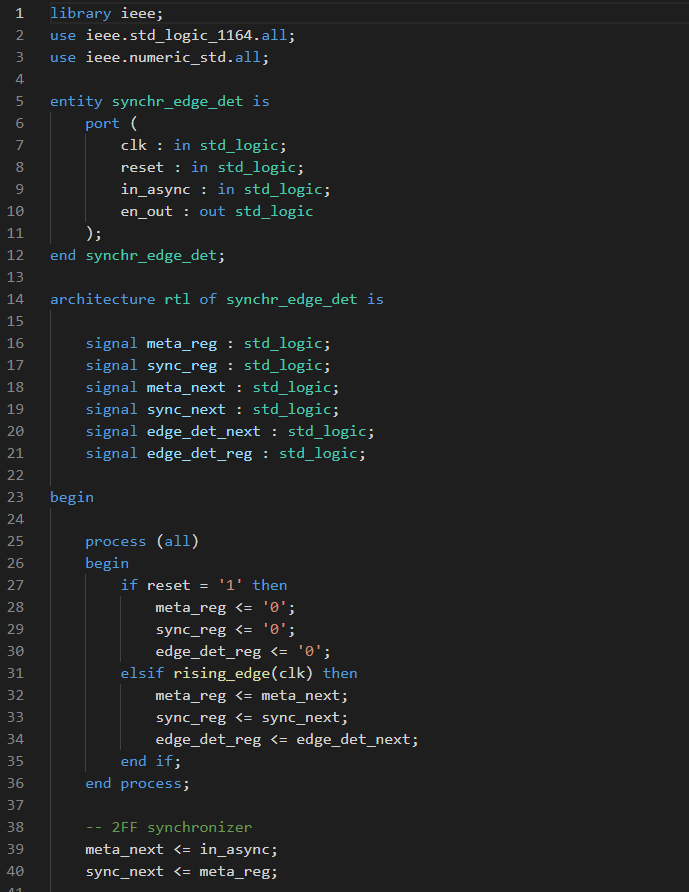
*top\_tb.vhd*

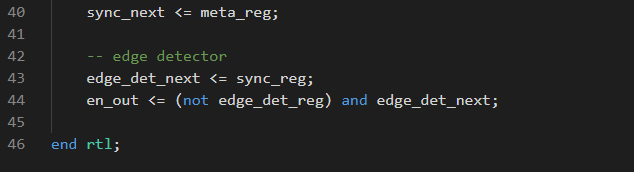




*top.vhd*

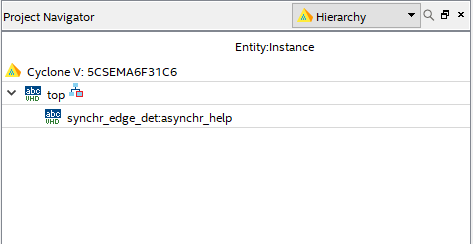
*synchr\_edge\_det.vhd*:

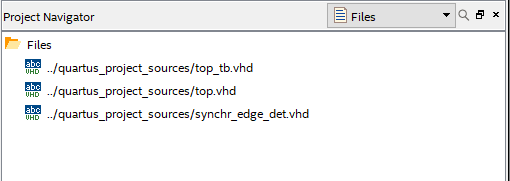




## D.

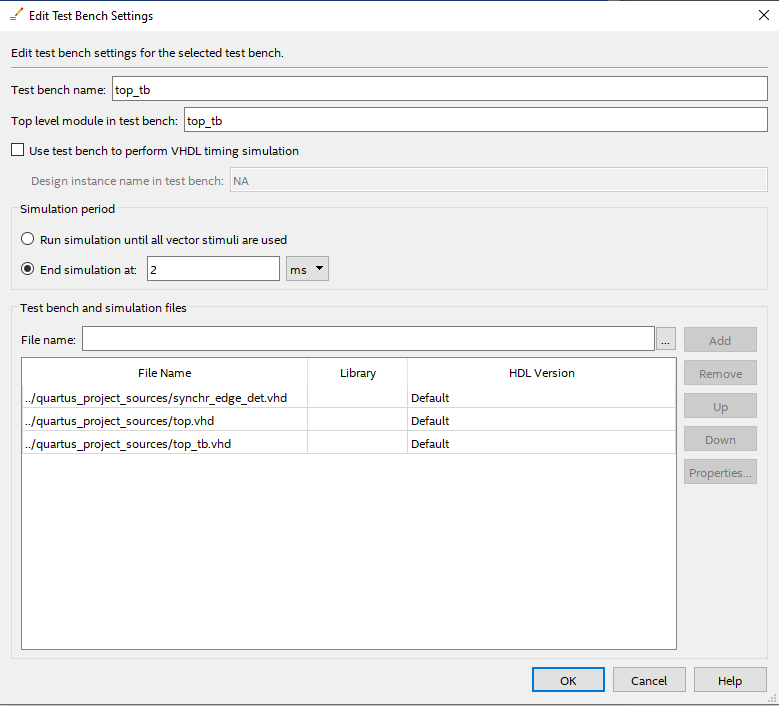
Projekta hierarhija un failu saraksts:





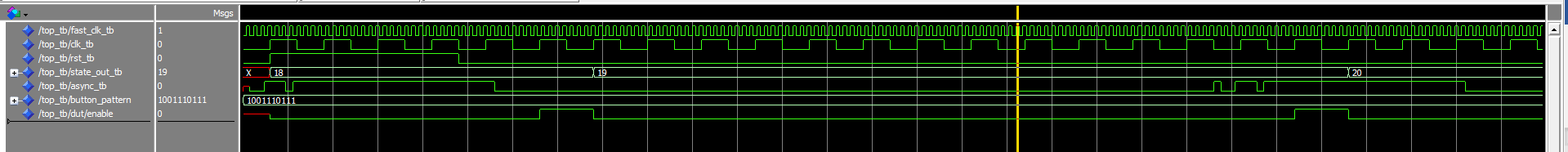
## E.

Quartus vidē testbench ir nokonfigurēts sekojoši:

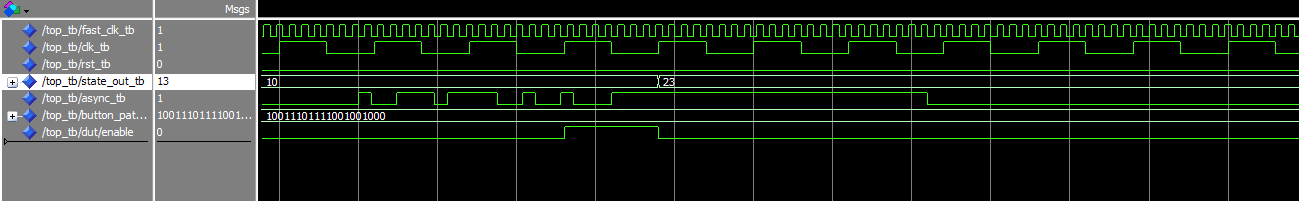


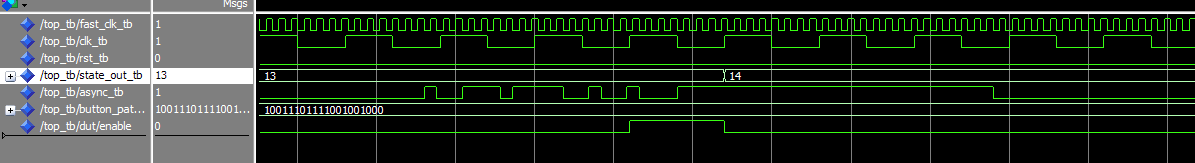
## F.

ModelSim vidē grafiskie rezultāti ir sekojoši, sākumā ir redzamā *reset* signāla darbība :



*Enable* signāla darbība, kopa ar kuru mainās testējamas secīgas shēmas stāvoklis. *Enable* signāls aktivizējas, kad asinhrona ieeja ir stabilizēta. Uz nākamo taktēšanas periodu mainās stāvoklis. Ir redzams ātrs taktēšanas signāls ar kuras palīdzību tiek ģenerēta asinhrona ieeja:





## Rezultātu analīze

Laboratorijas darba ietvaros bija uztaisīts asinhronas ieejas emulators (testbench iekšā) un sinhronizators kopā ar malas detektēšanas shēmu lai cīnīties ar metastabilitāti. Pēc simulācijas rezultātiem ir redzams , ka *enable* signāls netiek ģenerēts momentāni , bet pastāv sinhronizācijas aizture.