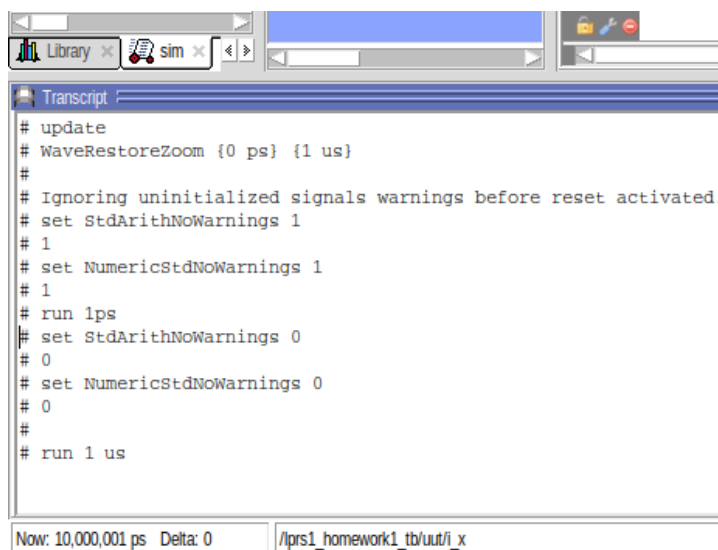


Упуство

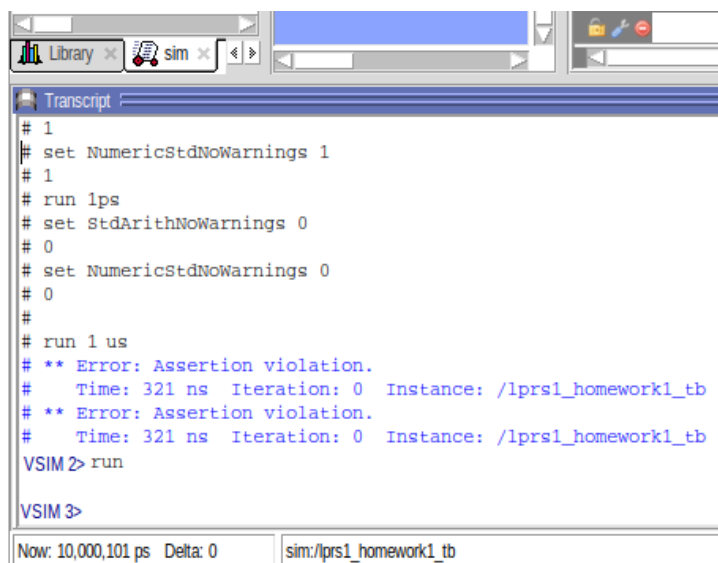
Потребно је реализовати дигитални систем састављен од комбинационих компоненти по следећем упуству.

1. На основу спецификације дате доле, нацртати блок шему система, по узору на Сliku 2 из Лабораторијске вежбе
3. Могуће је цртати ручно па сликати или пак у неком софтверу.
 - Сliku шеме сачувати под именом `block_diagram.jpg` у фолдеру `LPRS1_Homework1_RA_041_2019_Solution`.
2. Реализовати ову блок шему у VHDL-у. Реализацију урадити у `LPRS1_Homework1_RA_041_2019_Solution/lprs1_homework1.vhd` фајлу.
 - Изнад кода сваке компоненте у коментару написати име описане компоненте.
 - Напомена да су сви интерни сигнали 4-битни.
3. Проверити исправност решења путем симулације.
 - У пројекту вам је дат тестбенч који аутоматски проверава исправност решења. Потребно је само покренути симулацију. Ако је дизајн ваљан, Transcript панел у ModelSim-у ће бити без грешака, као на Сlici 1:



Слика 1: Симулација без грешака

Међутим, ако дизајн ниве ваљан, у Transcript панелу појавиће се грешке типа `Error: Assertion violation` као што је приказано на слици Сlici 2:



Слика 2: Симулација са грешакама

- Сам тестбенч није потребно мењати, нити ће исти бити прегледан. Он олакшава проверу и прегледање задатка.

- С друге стране дозвољено је мењање тестбенча, ради дебаговања и вежбања.
- Додатна необавезна могућност је коришћење емулятора. Да би се емулятор могао користити потребно је из *Lab2 projekat sa emulatorom* (фајл **Vezba2_Zad1.zip**) са веб странице предмета копирати **lprs1_emulator** фолдер у фолдер пројекта и покренути га како је већ описано у <https://www.youtube.com/watch?v=g1dg6uP2zj0>.

4. На крају, запаковати фолдер **LPRS1_Homework1_RA_041_2019_Solution** у zip фајл **LPRS1_Homework1_RA_041_2019_Solution .zip** и послати исти zip као решење свом асистенту преко чета у MS Teams-у.

Спецификација

Потребно је реализовати следећи систем:

1. На сигнал **s_shl** довести **i_x** померен 1 бит(а) у лево логички.
2. На сигнал **s_shr** довести **i_y** померен 3 бит(а) у десно аритметички.
3. На сигналу **s_dec** поставити бит са редним бројем **i_z** на јединицу а остале бите на логичку нулу.
4. Сигналу **s_add** доделити збир **s_shl** и **s_shr** сигнала.
5. Од **s_dec** одузети **i_x** и разлику доделити **s_sub** сигналу.
6. На сигнал **s_const0** доделити 2.
7. На сигнал **s_const1** доделити 11.
8. На сигнал **s_mux** доделити:
 - **s_sub** ако је **i_sel** једнако 0
 - **s_add** ако је **i_sel** једнако 1
 - **s_const1** ако је **i_sel** једнако 2
 - **s_const0** ако је **i_sel** једнако 3
9. Сигналу **o_res** доделити сигнал **s_mux**.
10. На сигнал **o_cmp(0)** довести логичку јединицу ако је **s_mux** различит од 0.
11. На сигнал **o_cmp(1)** довести логичку јединицу ако је **s_mux** већи од 0.
12. На сигнал **o_enc** довести индекс бита на логичкој јединици сигнала **s_mux**. Ако постоји више таквих бита, изабрати онај са најмањим индексом. Ако ни један бит није на логичкој јединици, резултат нека буде 3.