Универзитет у Београду
Електротехнички факултет
Катедра за електронику

Дигитална електроника 1

Лабораторијска вежба

Аритметичко логичка јединица (АЛУ)

Дизајн аритметичких и логичких кола за сабирање, одузимање и поређење бројева

- први део -

аутор: Марија Бежуљ

део материјала преузет са предмета Увод у пројектовање *VLSI* система

Београд, 2022.

1 Циљ вежбе

Циљ ове вежбе је да студенти генеришу VHDL фајлове за опис одговарајућег хардвера, упознају се са начином инстанцирања и коришћења компоненти, као и да науче да пишу VHDL код за тестирање и симулацију рада пројектованих кола.

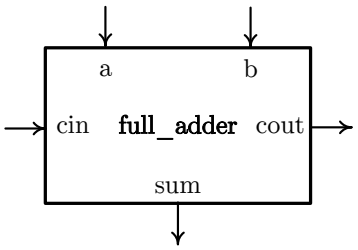
У оквиру ове вежбе потребно је направити нови пројекат, VHDL опис одговарајућих компоненти, а затим је потребно симулацијом пројектованих компоненти потврдити исправност реализованог дизајна.

На располагању је веб апликација *EDA Playground* за писање кода и симулацију, док се *testbench* фајлови могу генерисати помоћу *PERL* скрипте дате на линку.

Вежба је подељена на два дела: први део подразумева задатке које студенти треба даураде самостално код куће. Све фајлове из првог дела треба понети у лабораторију где ће студенти радити други део вежбе. У даљем тексту овог упутства описане су различите компоненте које је потребно уклопити у цео дизајн који ће бити описан у другом делу упутства, а који треба имплементирати у лабораторији.

2 Дизајн потпуног сабирача

Потребно је пројектовати једнобитни потпуни сабирач, са три улаза (сабирци и улазни пренос) и два излаза (збир и излазни пренос). Блок шема је приказана на слици 1.



Слика 1: Блок шема потпуног сабирача

Направити нови *playground* у *EDA Playground* веб апликацији. Искористити *design.vhd* фајл и у њему описати модул једнобитног потпуног сабирача, а *entity* назвати *full_ adder*.

У фајлу *testbench.vhd* napisati *testbench* којим треба испитати исправност рада потпуног сабирача. *Entity* у *testbench*-у nazvati *full_ adder_ tb*. Могуће је и генерисати *testbench* код помоћу *PERL* скрипте и едитовати га. Тестирати све могуће комбинације улаза.

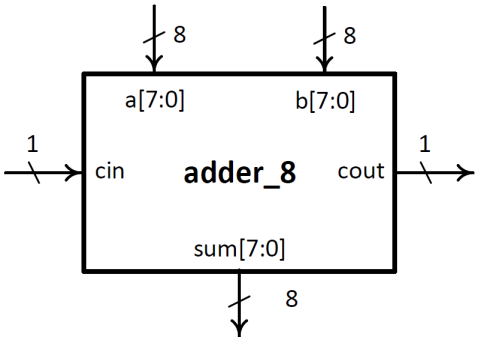
За симулацију селектовати алат *Aldec Riviera Pro 2022.04* из падајућег менија *Tools & Simulators*.

3 *8*-битни сабирач

Потребно је пројектовати осмобитни *ripple-carry* сабирач. Интерфејс овог сабирача приказан је на слици 2.

Фајл назвати *adder_ 8.vhd*. Креирање архитектуре овог сабирача извршити на структурном нивоу.

Креирати *test bench* фајл *adder_ 8_ tb.vhd* и написати или генерисати *test bench* код (преко *PERL* скрипте) и у оквиру њега испитати исправност рада 8-битног *ripple-carry* сабирача. Потребно је приказати неколико примера са прекорачењем и без прекорачења.



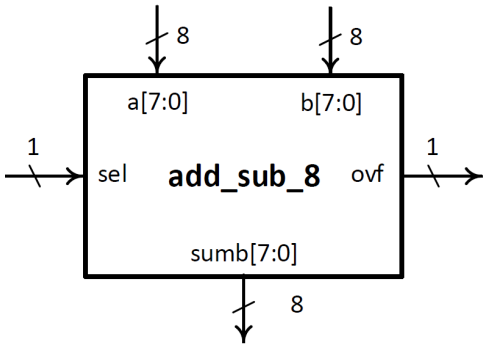
Слика 2: *8-битни сабирач*

4 *8*-битни сабирач/одузимач са overflow-ом

Пројектовати 8-битни сабирач/одузимач са overflow-ом (слика 3). Улази су 8-битни неозначени бинарни бројеви и сигнал *sel* који дефинише операцију:
$sel = 0$ - сабирање, $sel = 1$ - одузимање.
Излаз *sumb* је такође 8-битни неозначени бинарни број, док једнобитни сигнал ovf указује на појаву прекорачења приликом извођења захтеване операције над датим операндима.

Фајл назвати *add_ sub_ 8.vhd*, а *entity* назвати *add_ sub_ 8*. Креирање архитектуре овог модула извршити коришћењем реализованог 8-битног сабирача као компоненте и додавањем потребне логике.

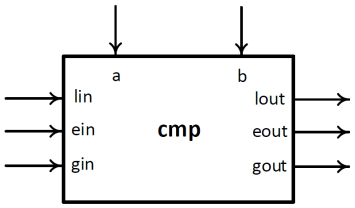
Креирати *test bench* фајл *add_ sub_ 8_ tb.vhd* и написати или генерисати *test bench* код (преко *PERL* скрипте) и у оквиру њега испитати исправност рада 8-битног сабирача/одузимача. Потребно је приказати неколико примера сабирања са прекорачењем и без прекорачења, као и одузимања са и без прекорачења.



Слика 3: 8-битни сабирач/одузимач са overflow-ом

5 Дизајн једнобитног компаратора

Потребно је пројектовати једнобитни компаратор, са пет улаза (бити које се пореде и улазни податак о односу претходних бита) и три излаза (излазни подаци о односу вредности). Блок шема је приказана на слици 4.



Слика 4: Блок шема једнобитног компаратора

Фајл назвати *cmp.vhd*, а *entity* назвати *cmp*. Описати архитектуру на нивоу понашања са циљем коришћења модула за креирање вишебитног компаратора (опис и пример реализације дати су у вежбама на табли - Кола средњег степена интеграције).

Креирати *test bench* фајл *cmp_ tb.vhd* и написати или генерисати *test bench* код (преко *PERL* скрипте) и у оквиру њега испитати исправност рада компаратора. Потребно је тестирати модул за све могуће комбинације улазних сигнала.

$$e_{out} = \overline{(a \oplus b)} \cdot e_{in}$$

$$l_{out} = \overline{a} \cdot b \cdot e_{in} + l_{in}$$

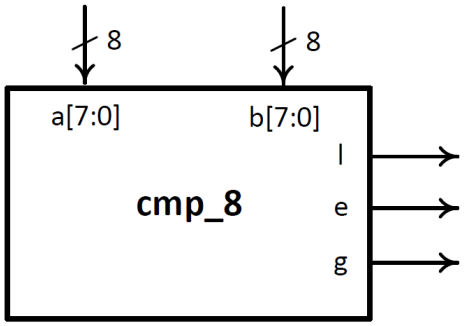
$$g_{out} = a \cdot \overline{b} \cdot e_{in} + g_{in}$$

6 *8*-битни компаратор неозначених бројева

Потребно је пројектовати 8-битни компаратор неозначених бројева (5). Излазни сигнали су једнобитни и указују да ли је број $a[7..0]$ мањи ($l = 1$), једнак ($e = 1$) или већи ($g = 1$) од броја $b[7..0]$.

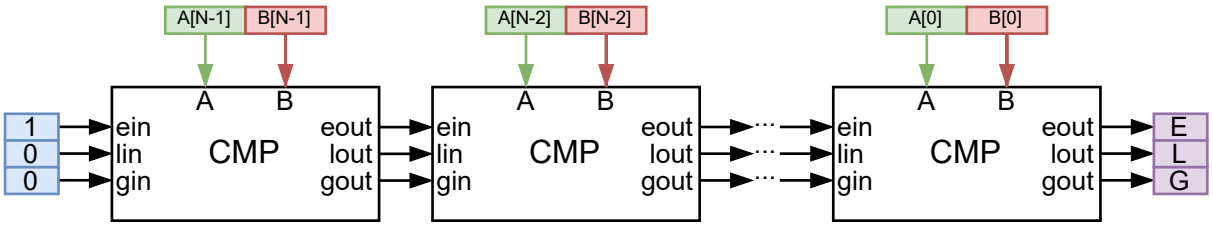
Фајл назвати *cmp_ 8.vhd*, а *entity* назвати *cmp_ 8*. Архитектуру 8-битног компаратора реализовати уланчавањем једнобитних компаратора (структурна архитектура). Водити рачуна да компонента има само улазе и излазе назначене на слици 5.

Креирати *test bench* фајл *cmp_ 8_ tb.vhd* и написати или генерисати *test bench* код (преко *PERL* скрипте) и у оквиру њега испитати исправност рада компаратора. Потребно је тестирати модул при различитим односима 8-битних улазних сигнала, показујући да сваки излазни сигнал исправно ради.



Слика 5: Компаратор 8-битних неозначених бројева

На слици 6 је приказана реализација осмобитног компаратора помоћу једнобитних компаратора.



Слика 6: Реализација компаратора