Универзитет у Београду
Електротехнички факултет
Катедра за електронику

Основи дигиталне електронике
- 13Е042ОД -

Лабораторијска вежба

Пројектовање хардвера на програмабилним компонентама коришћењем VHDL језика

Београд, мај 2016.

1. Циљ вежбе

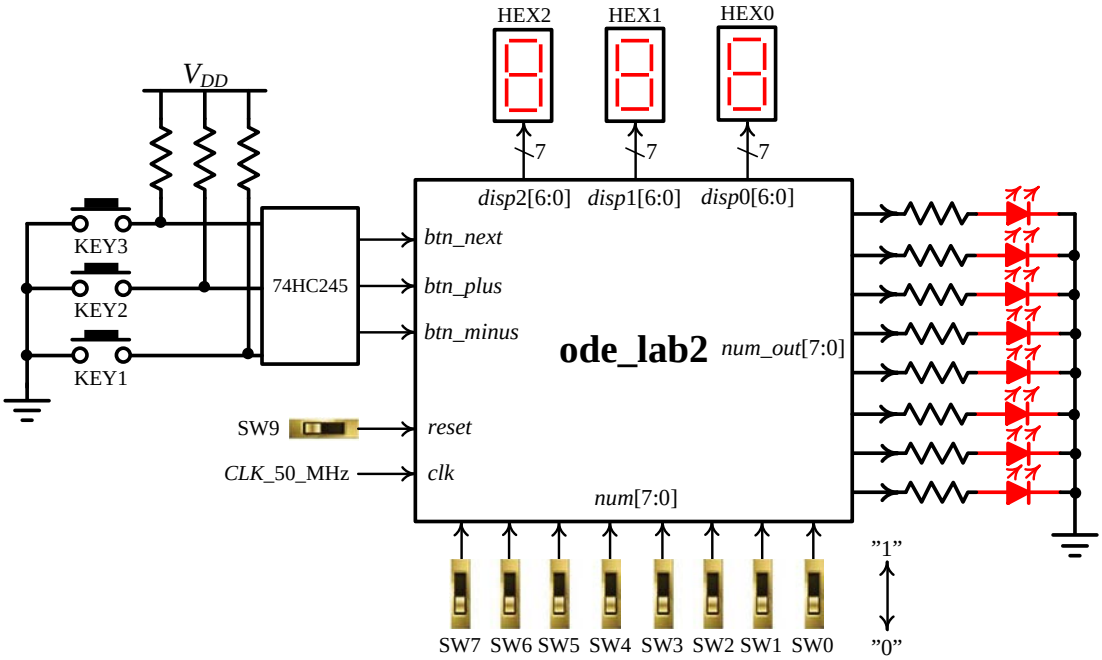
Циљ ове вежбе је да се студенти упознају са основним концептима пројектовања хардвера у VHDL језику. У вежби се користе алати *Altera Quartus II* и *ModelSim-Altera* чије се упутство за коришћење може наћи на сајту предмета.

У оквиру вежбе потребно је да студенти креирају неколико комбинационих кола и једноставну машину стања у VHDL језику за опис хардвера. Сваки од модула је потребно симулирати и на тај начин верификовати исправност рада. На крају се сви пројектовани модули заједно са додатним компонентама које су описане на крају овог упутства спајају у један јединствени систем који се синтетише и спушта на плочу са FPGA чипом. Плоча је *DE1-SoC* са FPGA чипом из *Altera Cyclone V SoC* фамилије.

2. Задатак вежбе

Све фајлове током ове лабораторијске вежбе чувати у директоријуму *E:\13E042OD\ode_lab2*.

Потребно је пројектовати дигитални систем **ode_lab2** који омогућава унос два 8-битна неозначена броја, а затим одабиром одговарајућег тастера извршава сабирање или одузимање датих бројева и резултат приказује у трајању од 3 секунде на 3 седмосегментна LED дисплеја. Систем се реализује на FPGA чипу који је залемљен на развојну плочу. На њој се налазе осцилатор који се користи као извор сигнала такта, тастери, прекидачи, LED дисплеји, LE диоде и друге компоненте које се неће користити у овој вежби. Блок шема система и интерфејс према наведеним компонентама на плочи су приказани на слици 1.

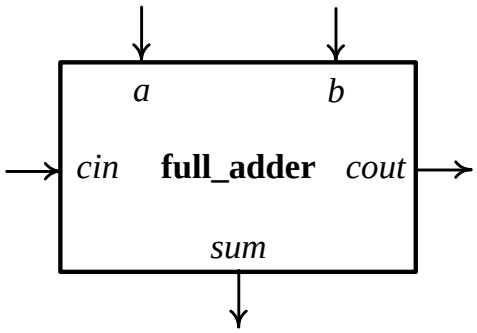


Слика 1. *Блок шема система и интерфејс према прекидачима, тастерима, дисплејима и LE диодама*

На почетку је систем у иницијалном стању и на LED дисплејима HEX0, HEX1 и HEX2 се приказују све три нуле. Притиском на тастер KEY3, на улазу *btn_next* се појављује дебаунсиран сигнал са тастера. Дебаунсирање сва три тастера ради посебан чип на плочи 74HC245. Појавом овог сигнала прелази се на одабир првог броја. Број се бира постављањем прекидача на плочи у одговарајући положај, а истовремено се на LED дисплејима приказује децимална вредност броја, а на LE диодама бинарна вредност броја. Поновним притиском на тастер KEY3 се прелази на одабир другог броја. Притиском на тастер KEY2, учитава се други број и на LED дисплејима се приказује децимална вредност збира учитаних бројева, а притиском на тастер KEY1 се учитава други број и на LED дисплејима се приказује децимална вредност разлике учитаних бројева. У оба случаја резултат се приказује 3 секунде, а затим се прелази у иницијално стање.

2.1. Пројектовање 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 2. Сви сигнали су типа ***std_logic***.



Слика 2. *Блок шема потпуног бројача*

Направити нови пројекат у *Altera Quartus II* 15 софтверу пратећи упутство за коришћење софтвера доступно на сајту предмета. Пројекат сместити у директоријум *E:\13E042OD\ode_lab2.* Фајл у коме се описује модул једнобитног потпуног сабирача назвати *full_adder.vhd*, а *entity* назвати *full_adder*. Никако не називати модуле и фајлове именима „*proba*“, „*prvi_fajl*“ и слично! Не заборавити да увек модул који се тестира треба да буде постављен за *Top Level Entity*.

Креирати *test bench* фајл *full_adder_tb.vhd* или генерисати и едитовати *test bench* фајл *full_adder.vht*, којим треба испитати исправност рада потпуног сабирача.

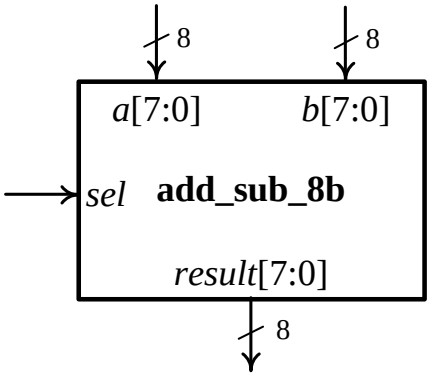
За симулацију користити алат *ModelSim-Altera* према упутству за коришћење софтвера са сајта.

Позвати дежурног асистента да верификује успешно комплетирање овог дела вежбе!

2.2. Пројектовање 8-битног сабирача и одузимача

Потребно је пројектовати комбинациони модул који на улазу има два осмобитна броја *a*[7:0] и *b*[7:0] типа ***std_logic_vector*** и један селекциони сигнал *sel* типа ***std_logic***, а на излазу има један осмобитни број *result*[7:0] типа ***std_logic_vector*** (Напомена: у

даљем тексту се подразумева да су сви улазни и излазни сигнали свих модула типа ***std_logic*** или ***std_logic_vector***). Блок шема је приказана на слици 3. У оквиру овог модула потребно је реализовати 8-битни *ripple carry* сабирач истанцирањем 8 једнобитних сабирача из претходне тачке (препорука: користити наредбу ***for i in 7 downto 0 generate***), при чему се на улазе једнобитних сабирача доводе одговарајући сигнали којим се омогућава да модул ради сабирање ако је улазни селекциони сигнал *sel* = ’0’ или одузимање ако је *sel* = ’1’.



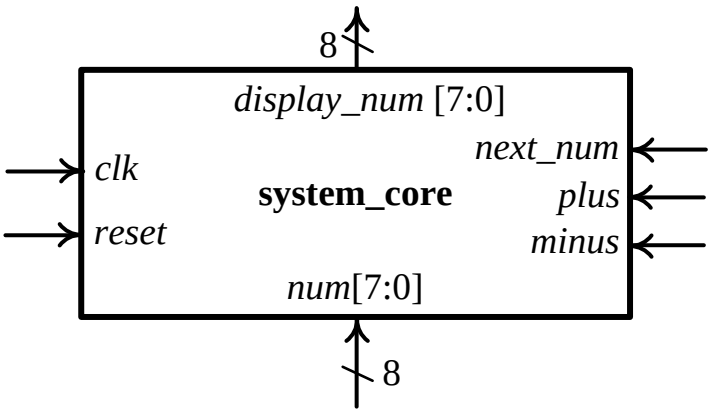
Слика 3. *Блок шема* 8-*битног сабирача и одузимача*

Фајл у коме се описује овај модул назвати *add_sub_8b.vhd*, а *entity* назвати *add_sub_8b*. Креирати *test bench* фајл *add_sub_8b_tb.vhd* или генерисати и едитовати *test bench* фајл *add_sub_8b.vht*, а затим симулацијом верификовати исправност рада овог модула.

<u>Позвати дежурног асистента да верификује успешно комплетирање овог дела вежбе!</u>

2.3. Пројектовање машине стања и језгра система

Језгро система ***system_core*** се састоји од синхроне машине стања којом се контролише рад система, компоненте сабирача/одузимача из претходне тачке, локалних регистара и околне логике. Језгро система на улазу има 8-битни број *num*[7:0] који представља број који је потребно учитати у одговарајућим стањима у локалне сигнале. Цео систем се ресетује сигналом *reset*, активним са логичком нулом, а синхрони делови раде на сигнал такта *clk*. Блок шема система је приказана на слици 4.

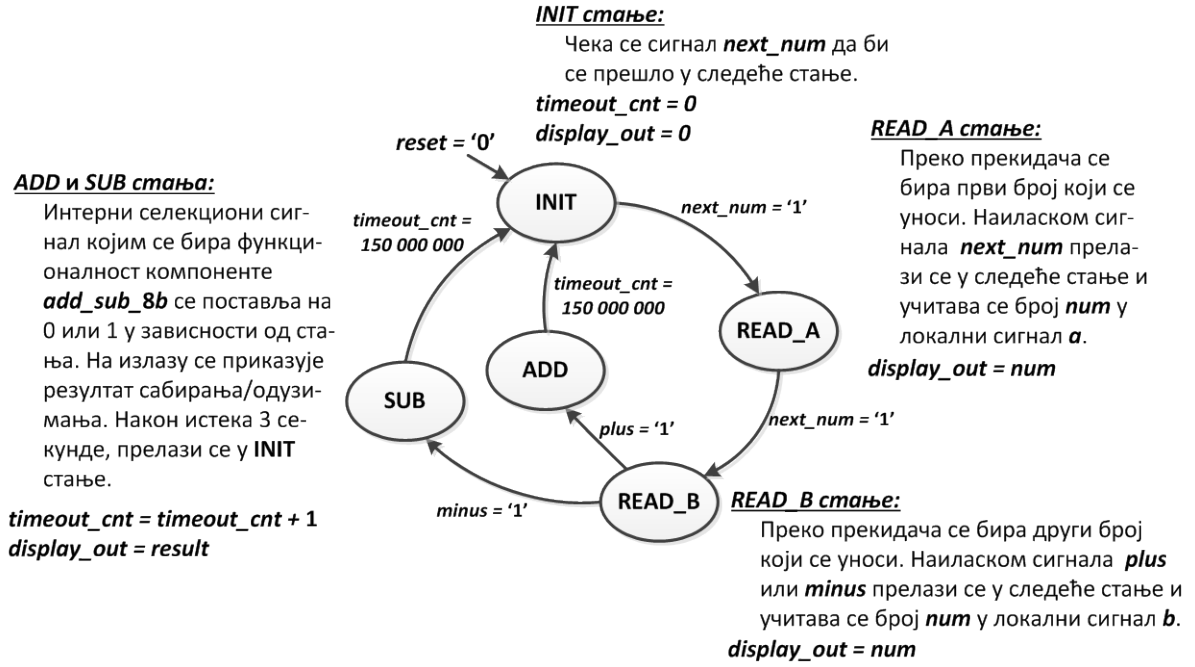


Слика 4. *Блок шема главног дела система*

Улазни сигнали *next_num*, *plus* и *minus* су контролни сигнали који одређују прелазе између стања у машини стања. Сматрати да ни један од наведених сигнала не траје дуже од једне периоде сигнала такта. Излазни број *display_num*[7:0] је број који се коришћењем додатних модула приказује на LED дисплеју. Машина стања коју је потребно реализовати је приказана на слици 5.

Ако је активан сигнал *reset*, систем се налази у стању *INIT*. У овом стању, потребно је да излазни број *display_num*[7:0] буде једнак нули. По деактивирању сигнала *reset*, систем остаје у стању *INIT* све док сигнал *next_num* не добије вредност ’1’. Тада се на прву следећу узлазну ивицу сигнала такта прелази у стање *READ_A*. Ово је стање у коме се врши одабир првог броја. С обзиром на то да је улазни број дефинисан стањима прекидача на плочи, потребно је само проследити његову вредност на излаз *display_num*[7:0] како би се, након пропагације кроз додатна кола, на LED дисплејима видела свака промена улазног броја. Када сигнал *next_num* поново добије вредност ’1’, потребно је да се на прву следећу узлазну ивицу сигнала такта пређе у стање *READ_B*. Такође, у истом тренутку је потребно и учитати податак са улаза у локални сигнал *a*[7:0] у ком се чува вредност првог броја. У стању *READ_B* је, као и у претходном стању, потребно омогућити да се на *display_num*[7:0] прослеђује улазни податак. У случају да сигнал *plus* добије вредност ’1’, прелази се у стање *ADD*, а у случају да сигнал *minus* добије вредност ’1’, прелази се у стање *SUB*. Такође, у истом тренутку је потребно и учитати податак са улаза у локални сигнал *b*[7:0] у ком се чува вредност другог броја.

Учитавање података *a*[7:0] и *b*[7:0] је најједноставније урадити у посебном секвенцијалном процесу у коме се проверава да ли се систем налази у одговарајућем стању за учитавање и да ли улазни сигнали имају одговарајуће вредности.



Слика 5. *Машина стања главног дела система*

У стањима *ADD* и *SUB* потребно је да излаз *display_num*[7:0] има вредност резултата из компоненте *add_sub_8b*. У овим стањима се остаје 3 секунде, а по истеку тог времена, потребно је прећи у почетно *INIT* стање. Мерење времена је најлакше реализовати

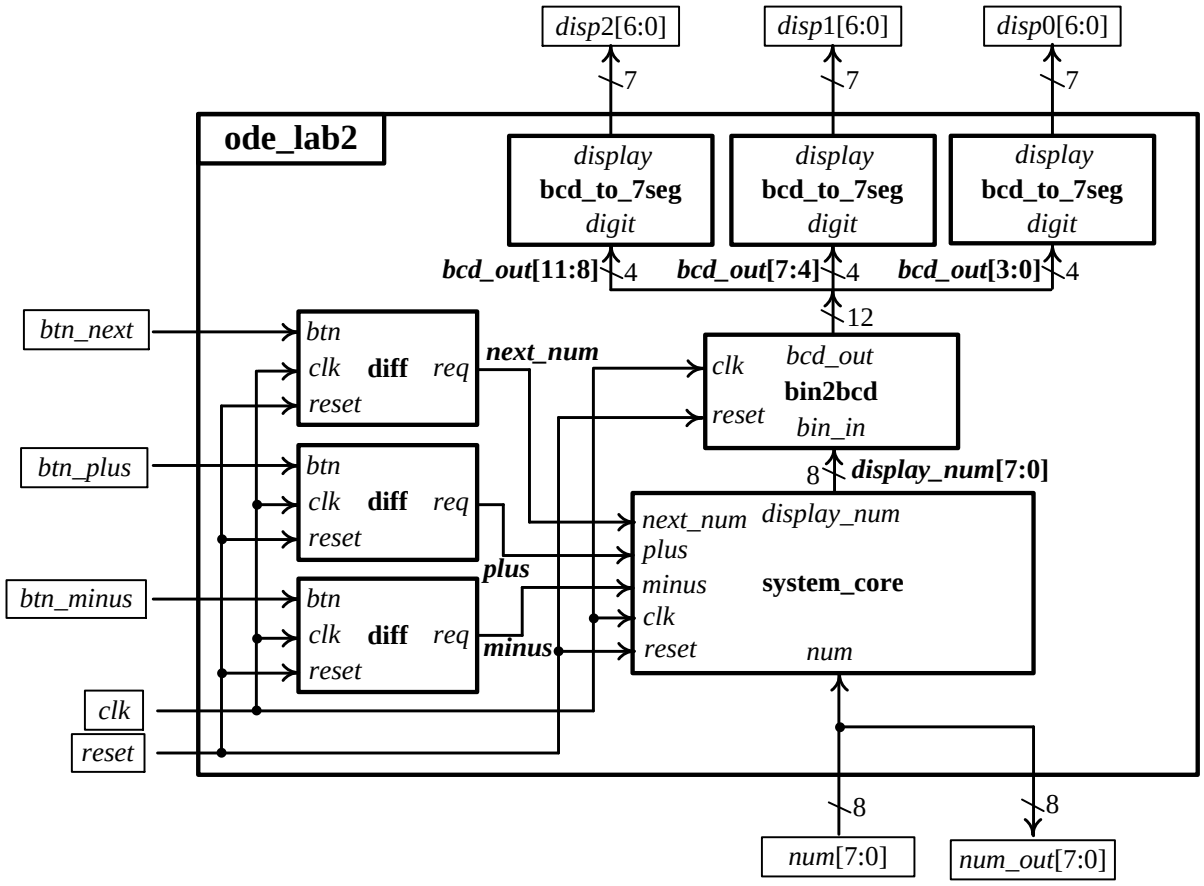
бројачем који броји до 150000000 (учестаност такта је 50 MHz). Када је вредност бројача једнака 150000000, прелази се у почетно стање у коме се вредност бројача ресетује на 0. Бројач реализовати као посебан секвенцијани процес у коме се у зависности од стања система, на сваку узлазну ивицу мења вредност бројача која је представљена локалним сигналом ***timeout_cnt*** типа ***integer***. За потребе симулације, подесити да бројач броји до 10 у наведеним стањима, а пре спуштања на плочу на крају вежбе вратити вредности на 150000000.

Фајл у коме се описује овај модул назвати *system_core.vhd*, а *entity* назвати *system_core*. Креирати *test bench* фајл *system_core_tb.vhd* или генерисати и едитовати *test bench* фајл *system_core.vht*, а затим симулацијом верификовати исправност рада машине стања и осталих делова модула.

Позвати дежурног асистента да верификује успешно комплетирање овог дела вежбе!

2.4. Повезивање целог система

У новом VHDL фајлу *ode_lab2.vhd* креирати модул *ode_lab2* чија је блок шема приказана на слици 6. На шеми су наведени сви улазни и излазни сигнали који треба да се налазе у интерфејсу овог модула. Систем реализовати инстанцирањем модула пројектованог у претходној тачки и модула који су доступни у фајловима *diff.vhd*, *bin2bcd.vhd* и *bcd_to_7seg.vhd*, а који прате ово упутство. Најпре је неопходно додати све ове фајлове у пројекат, а онда инстанцирати модул *bin2bcd*, три модула *diff* и три модула *bcd_to_7seg*. Повезати сваку појединачну компоненту коришћењем помоћних сигнала назначених на блок шеми.



Слика 6. *Блок шема целог система*

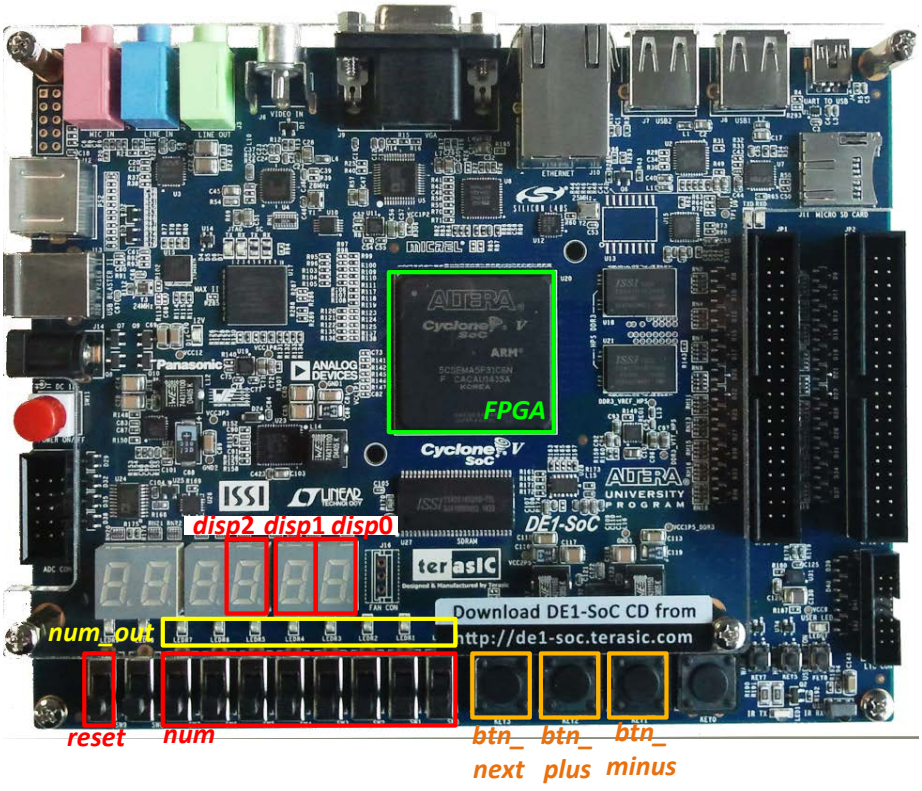
Модул *diff* представља детектор притиска тастера који ради додатно дебаунсирање улазног сигнала и на излазу генерише логичку јединицу у трајању само једне периоде сигнала такта. Детаљи машине стања која реализује овај модул дати су у додатку 3.1.

С обзиром на то да је неопходно да се 8-битни бинарни број прикаже на 3 седмосегментна LED дисплеја и то на сваком дисплеју по једна децимална цифра, потребно је одредити ове децималне цифре. Ово се може урадити новом машином стања која рекурзивно дели број са 10 и посматра остатке при дељењу, али је хардверско дељење обично захтеван процес по питању ресурса и времена извршавања. У овој вежби је одабран други приступ где се бинарни 8-битни број пребацује у број у природном BCD коду где свака суседна 4 бита представљају једну цифру. Машина стања која реализује ову конверзију описана је у додатку 3.2, а реализована је у модулу *bin2bcd*. На излазу овог модула, када се заврши процес конверзије, добија се 12-битни податак који представља 3 цифре.

Да би се израчунате цифре приказале на LED дисплејима, неопходно је реализовати комбинациону мрежу која 4-битни бинарни број пребацује у кôд 7 сегмената. Ова комбинациона мрежа је реализована у модулу *bcd_to_7seg*.

Након повезивања система и успешне компилације, позвати дежурног асистента да помогне у спуштању пројектованог дизајна на плочу.

Користећи .tcl скрипту из прилога овог упутства и пратећи упутсво за коришћење софтвера, мапирати сигнале на одговарајуће пинове и испрограмирати FPGA чип. Тестирати пројектовани дизајн на развојном систему. *DE1-SoC* развојни систем са обележеним периферијама које су повезане на обележене сигнале пројектованог модула приказан је на слици 7.



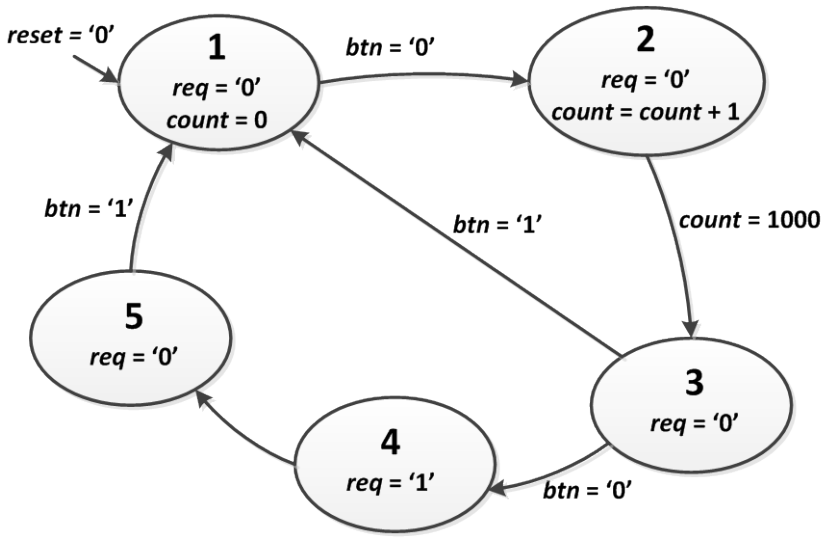
Слика 7. DE1-SoC развојни систем са обележеним улазним и излазним периферијама

3. Додаци

У овом поглављу су додатно описане компоненте које се не реализују у вежби, али се користе за реализацију целог система. Ова објашњења и кодови могу послужити студентима за боље разумевање реализације машина стања у VHDL-у.

3.1. Детектор притиска тастера

Машина стања којом се реализује детекција притиска тастера је приказана на слици 8. Иницијално стање је стање 1, а њему се остаје све док се не деси притисак тастера (сигнал *btn* добија вредност ’0’ – приметити са слике 1 да су тастери активни са логичком нулом). У стању 2 се остаје све док интерни бројач не изброји до 1000 и тада се прелази у стање 3. Овде се даје времена улазном сигналу да се стабилизује, а затим се у стању 3 провери да ли је сигнал и даље активан. Ако није, значи да се само десила нека сметња и да тастер није притиснут, па се систем враћа у иницијално стање. У случају да је сигнал *btn* и даље на ’0’ прелази се у стање 4 које траје један тактни циклус и у том стању излаз *req* добија вредност ’1’. Из стања 4 се безусловно прелази у стање 5 у коме машина стања остаје све док улазни сигнал не постане неактиван.



Слика 8. *Машина стања детектора притиска тастера*

3.2. Конвертор бинарног броја у број у природном BCD коду

Алгоритам конверзије се састоји од следећих корака:

1) Ако је било која цифра (стотине, десетице, јединице) већа од 4 додати 3 на ту цифру
2) Померити све бите за једно место улево
3) Ако је већ обављено 8 померања процес конверзије је завршен
4) Поновити корак 1.

Пример конверзије броја 178 (10110010) из бинарног у BCD коришћењем горе описаног алгоритма је приказан на слици 9.

shift_cnt	shift_reg				
7	0000	0000	0000	10110010	
6	0000	0000	0001	01100100	
5	0000	0000	0010	11001000	
4	0000	0000	0101	10010000	
4	0000	0000	1000	10010000	*корекција!*
3	0000	0001	0001	00100000	
2	0000	0010	0010	01000000	
1	0000	0100	0100	10000000	
0	0000	1000	1001	00000000	
0	0000	1011	1100	00000000	*корекција!*
0	0001	0111	1000	00000000	
	1	**7**	**8**		

Слика 9. *Пример конверзије из бинарног у BCD формат*

Описани алгоритам се може једноставно имплементирати коришћењем коначне машине стања. Из алгоритма се јасно издвајају 3 стања: почетно стање (*Idle*), стање померања (*Shift*) и стање корекције, односно сабирања (*Add*).

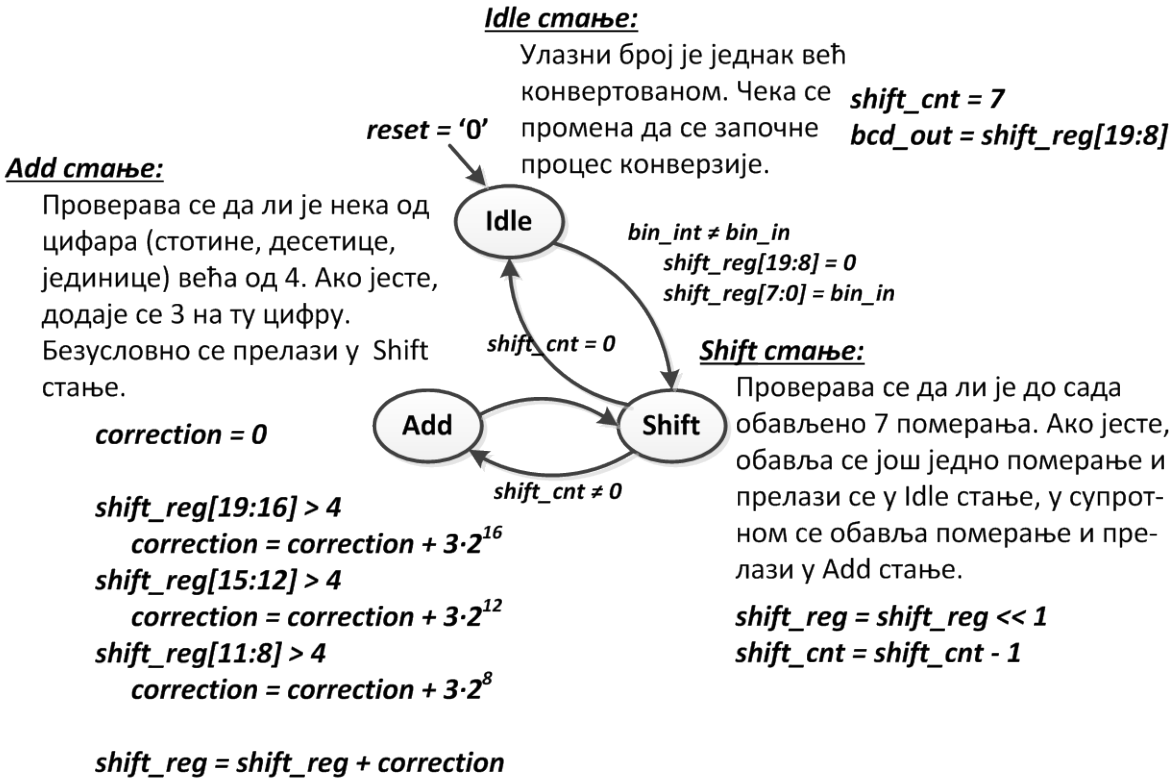
Машина стања се налази у почетном стању док год је улазни бинарни број једнак броју уписаном у интерни регистар који представља бинарну вредност тренутног BCD излаза. У тренутку када се детектује да се бинарни број који се налази у интерном регистру и улаз разликују значи да излаз више није валидан па је потребно започети нови процес конверзије иницијализовањем свих интерних сигнала и преласком у стање померања.

У стању померања је потребно проверити да ли је до сада обављено 7 померања. Ако јесте обавља се још 1 померање чиме је процес конверзије завршен и прелази се у почетно стање. У супротном се обавља померање, бројач преосталих померања се умањује за 1 и прелази се у стање корекције.

У стању корекције проверава се вредност сваке од цифара и додаје одговарајући фактор на корекциони члан. Корекциони члан је потребно дефинисати унутар самог стања корекције пошто он представља променљиву локалног карактера. Обратити пажњу на коришћење променљиве у коду из прилога упутства.

Архитектура описане машине стања приказана је на слици 10.

Обратити пажњу у кôду да се излазни BCD број мења у секвенцијалном процесу јер је потребно да његова вредност остане непромењена у току конверзије, а мења се тек на крају конверзије.



Слика 10. *Машина стања којом се реализује конверзија* 8*-битног бинарног броја у BCD формат*