АКАДЕМИЈА ТЕХНИЧКО-УМЕТНИЧКИХ СТРУКОВНИХ СТУДИЈА БЕОГРАД

ОДСЕК ВИСОКА ШКОЛА ЕЛЕКТРОТЕХНИКЕ И РАЧУНАРСТВА

**Ђурић Маша**

**РЕАЛИЗАЦИЈА САМОУСЛУЖНИХ АПАРАТА ПРИМЕНОМ МАШИНЕ СТАЊА И ИМПЛЕМЕНТАЦИЈА У ПРОГРАМАБИЛНОЈ ЛОГИЦИ**

**- завршни рад -**



Београд, јун 2023.

Кандидат: **Ђурић Маша**

Број индекса: **елите-6/19**

Студијски програм: **Електроника и телекомуникације**

Тема: **Реализација самоуслужних апарата применом машине стања и имплементација у програмабилној логици**

Основни задаци:

**1.** **Опис рада самоуслужног апарата применом машине стања**

**2. Опис машине стања HDL језиком**

**3. Имплементација дизајна у програмабилном логичком колу и тестирање**

Ментор:

Београд, јун 2023 годинe.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

др Драгана Прокин, проф.ВИШЕР

резиме:

Циљ завршног рада је био опис машине стања самоуслужног апарата за кафу. У првом делу дат је преглед и опис рада машина стања, као опис самоуслужног апарата за кафу. У другом делу рада је представљен опис машине стања применом VHDL (Very High-Speed Integrated Circuit Hardware Description Language) језика. На крају, у трећем делу завршног рада, описана је тест шема за успешну симулацију и проверу исправности VHDL описа дизајна самоуслужног апарата за кафу. Резултати симулације су потврдили успешну имплементацију.

**Кључне речи:** машина стања, HDL (Hardware Description Language) језик, VHDL (Very High-Speed Integrated Circuit Hardware Description Language) језик, тест шема, симулација

ABSTRACT:

The goal of the final paper was to describe the state machine of a self-service coffee machine. The first part provides an overview and description of the operation of state machines, specifically focusing on the description of a self-service coffee machine. The second part of the thesis presents the description of the state machine using VHDL (Very High-Speed Integrated Circuit Hardware Description Language). Finally, in the third part of the final thesis, a test scheme is described for successful simulation and verification of the VHDL description of the self-service coffee machine design. The simulation results confirmed the successful implementation.

**Key words**: State machine, HDL (Hardware Description Language), VHDL (Very High-Speed Integrated Circuit Hardware Description Language), test scheme, simulation.

САДРЖАЈ:

[1. УВОД 1](#_Toc137667032)

[2. ОПИС РАДА САМОУСЛУЖНОГ АПАРАТА ПРИМЕНОМ МАШИНЕ СТАЊА 2](#_Toc137667033)

[2.1. Машина стања 2](#_Toc137667034)

[2.1.1. Мурова машина стања 5](#_Toc137667035)

[2.1.2. Милијева машина стања 5](#_Toc137667036)

[2.1.3. Репрезентација машина стања 6](#_Toc137667037)

[2.2. Самоуслужни апарати (апарати за кафу) 9](#_Toc137667038)

[2.2.1. Историјат самоуслужних апарата 10](#_Toc137667039)

[2.3. Дијаграм машине стања самоуслужног апарата за кафу 10](#_Toc137667040)

[3. ОПИС МАШИНЕ СТАЊА HDL ЈЕЗИКОМ 13](#_Toc137667041)

[3.1. VHDL 13](#_Toc137667042)

[3.1.1. Структура VHDL фајла 14](#_Toc137667043)

[3.1.2. Основни елементи VHDL језика 17](#_Toc137667044)

[3.2. Имплементација у VHDL језику 19](#_Toc137667045)

[3.2.1. Архитектура 25](#_Toc137667046)

[4. ИМПЛЕМЕНТАЦИЈА ДИЗАЈНА У ПРОГРАМАБИЛНОМ ЛОГИЧКОМ КОЛУ И ТЕСТИРАЊЕ 34](#_Toc137667047)

[4.1. Детектор успонске ивице и његова примена на тест шеми 34](#_Toc137667048)

[4.2. Тест шема самоуслужног апарата 36](#_Toc137667049)

[4.3. Симулација тест шеме самоуслужног апарата 38](#_Toc137667050)

[5. ЗАКЉУЧАК 44](#_Toc137667051)

[6. ИНДЕКС ПОЈМОВА 45](#_Toc137667052)

[7. ЛИТЕРАТУРА 47](#_Toc137667053)

[8. изјава о академскoј честитости 48](#_Toc137667054)

# УВОД

Главни циљ овог завршног рада јесте реализација самоуслужног апарата применом машине стања и имплементација у програмабилној логици. Изабрано је да се опише самоуслужни апарат за кафу, јер ови уређаји имају широку примену.

До описаног решења дошло се анализом логике на којој се базира рад самоуслужних апарата, а затим је дефинисана спецификација самоуслужног апарата за кафу и описана реализација применом машине стања. Предложеним решењем требало је решити проблем понашања аутомата, ако не постоје потребни састојци за израду напитка и уколико не постоји могућност повратка кусура. Ова проблематика је описана у првом поглављу рада „Машина стања“.

Машина стања је описана помоћу VHDL језика. Разлог за избор овог језика наспрам других HDL језика је његова читљивост, шира примена и генерално лакша реализација потребних захтева. Опис машине стања HDL језиком, врсте HDL језика, структура VHDL језика и опис машине стања VHDL језиком, као и код објашњени су у другом поглављу под називом „Опис машине стања HDL језиком“.

Тестирање дизајна применом тест шеме и симулације описано је у трећем поглављу завршног рада под називом „Имплементација дизајна у програмабилном логичком колу и тестирање.

# ОПИС РАДА САМОУСЛУЖНОГ АПАРАТА ПРИМЕНОМ МАШИНЕ СТАЊА

## Машина стања

Машине стања (Finite State Machine, FSM) припадају групи секвенцијалних мрежа. Машина стања као модел заправо може да моделује произвољну секвенцијалну мрежу. Основна разлика између машина стања и осталих врста секвенцијалних мрежа огледа се у томе да машине стања имају нерегуларну структуру функције наредног стања. Флип-флопови, регистри, меморије и бројачи су секвенцијалне мреже са регуларном структуром функције наредног стања.

Формално, машине стања се дефинишу помоћу пет објеката:

* Скуп симболичких стања у којима се може наћи
* Скуп улазних сигнала на које је осетљива
* Скуп излазних сигнала које генерише
* Функције наредног стања, која се користи за одређивање наредног стања у коме ће се наћи машина
* Излазне функције, која се користи за одређивање наредних вредности излазних сигнала које генерише машина.

Стање машине представља јединствено унутрашње стање у коме се машина може наћи. Скуп свих стања машине мора бити коначан и како време тече стања се мењају, прелазе из тренутног стања у следеће стање. Наредно, следеће стање одређује се функцијом наредног стања, која на основу текућег стања у којем се налази машина и текућих вредности улазних сигнала одређује наредно стање у коме ће се наћи машина. У случају синхроних машина стања, прелази су контролисани помоћу сигнала такта, где се стање мења на успонској ивици сигнала такта.

На основу улазних сигнала или излазне функције машине стања одређујемо вредности излазних сигнала машине. У зависности од улазних сигнала ове функције, излазне сигнале ове машине делимо у две велике групе:

* Мурове излазе
* Милијеви излази

Машине стања које садрже само излазе Муровог типа називају се Мурове машине стања, док се машине стања које садрже само излазе Милијевог типа називају Милијеве машине стања. Иако постоје машине стања које су стриктно Муровог или Милијевог типа, у пракси се најчешће срећу машине стања код којих имамо и једну и другу врсту излаза.

Diagram

Description automatically generated

Слика 2.1.1 – Генерална структура машине стања

Машине стања у општем случају поседују следеће портове:

* N улазних портова,
* M1 излазних портова Милијевог типа,
* M2 излазних портова Муровог типа,
* clk улаз за синхронизацију рада коначног аутомата

Опционо, машине стања могу поседовати и следеће портове:

* reset улазни порт за иницијализацију садржаја регистра стања
* clock enable улазни порт за дозволу рада машине стања на узлазној ивици такта.

N улазних портова машине стања може бити организован у N једнобитних улазних портова, али они такође могу бити и вишебитни улазни портови. Принцип рада машине стања је следећи: наиласком растуће ивице сигнала такта машина стања, у зависности од стања у коме се тренутно налази, анализира тренутно стање на улазним портовима и, користећи функцију наредног стања, одређује наредно стање у коме ће се наћи током следеће периоде clk сигнала. Истовремено, машина стања генерише нове вредности за одговарајуће излазе Милијеве или Мурове машине, на основу тренутног стања у коме се налази и тренутних вредности сигнала улазних портова. Слично као код улазних сигнала, M1 Милијевих и M2 Мурових излазних портова може бити једнобитно или вишебитно.

Diagram, engineering drawing

Description automatically generated

Слика 2.1.2 – Временски диаграм машине стања са једним улазом и два излаза

Са слике 2.1.2 можемо видети да се машина стања пре наиласка прве растуће ивице clk сигнала (у тренутку t1) налази у стању S1. Муров излаз (moore\_out сигнал) зависи само од тренутног стања у коме се машина стања налази и можемо видети да је на високом логичком нивоу. То значи да ће машина стања овај излаз увек постављати на високи логички ниво када се налази у стању S1. Милијев излаз зависи не само од стања у коме се машина тренутно налази већ и од тренутне вредности улаза, у случају са слике 2.1.2 улаза input. Са слике 2.1.2 можемо видети да излаз Милијеве машине (mealy\_out сигнал) прати таласни облик улазног сигнала input и не зависи од стања у коме се машина налази. Наиласком растуће ивице clk сигнала у тренутку t1 машина стања прелази у ново стање, S2, у коме ће остати све до наиласка следеће растуће ивице, у тренутку t2. Када машина пређе у стање S2 долази до промене на Муровом излазу (сигнал moore\_out постаје 0), што значи да машина стања са слике 2.1.2 сваки пут када се нађе у стању S2 спушта moore\_out излаз на 0. Карактеристичне промене до којих долази приликом рада машине на слици 2.1.2 означене су различитим бојама. Црвеном бојом означени су тренуци промене стања у коме се машина налази. Плавом бојом означени су тренуци када долази до промене на Муровом излазу. Зеленом бојом означени су тренуци када долази до промена на Милијевом излазу.

Са слике 2.1.2 може се уочити и једна од главних разлика између Милијевих и Мурових излаза. Милијеви излази се генеришу брже од Мурових, чим дође до промене на неком од улаза који контролише генерисање Милијевог излаза. Међутим, свака промена на неком од контролних улаза, резултује у промени одговарајућег Милијевог излаза, што значи да ће се и сви гличеви који су евентуално присутни на неком од улаза пренети и на Милијев излаз, што може представљати проблем. Са друге стране, Мурови излази се мењају само приликом наиласка растуће ивице сигнала, што значи да уносе кашњење од једне периоде clk сигнала, али се на њима не могу појавити гличеви. Уколико нам је потребна брза реакција машине стања на промене улазних сигнала, потребно је користити Милијеве излазе. Уколико су нам потребни излазни сигнали машине стања без гличева, потребно је користити Мурове излазе.

### Мурова машина стања

Мурова машина стања се дефинише скупом стања у којима се машина може наћи, скупом функција прелаза из стања у стање, као и скупом излаза које ће генерисати машина за свако стање.

Код Мурове машине тренутни излаз из машине је функција само стања у коме се у том тренутку налази машина, независно од тренутног стања на улазу машине.

Diagram

Description automatically generated

Слика 2.1.1.1 – Мурова машина стања

### Милијева машина стања

За разлику од Мурове машине стања, Милијева машина стања се дефинише као скуп стања у којима се машина може наћи, функцијама прелаза између стања, која за разлику од Мурове машине зависе и од стања у коме се машина тренутно налази, као и скупом излаза.

Разлика између Мурове и Милијеве машине стања јесте у томе што је код Мурове машине стања тренутни излаз из машине искључиво одређен стањем у коме се машина налази у том тренутку, док код Милијеве машине стања излаз је одређен тренутним стањем машине као и стањем на улазу машине.

Diagram

Description automatically generated

Слика 2.1.2.1 – Милијева машина стања

### Репрезентација машина стања

Приликом спецификације машине стања, што је први корак у њиховом моделовању помоћу VHDL језика, могу се користити различити модели репрезентације. Следећа три модела се најчешће користе у пракси:

* текстуални опис
* опис помоћу таблице прелаза/излаза
* опис помоћу дијаграма стања

#### Текстуални опис

Текстуални опис као модел за спецификацију жељеног понашања машине стања често се користи у пракси иако има озбиљне недостатке. Главни недостатак овог приступа огледа се у недовољној прецизности спецификације и могућим различитим интерпретацијама. Ово је последица превелике слободе изражавања коју нам пружају природни језици (српски, енглески). Једна те иста реченица може се протумачити на различите начине од стране различитих људи. Ово постаје још већи проблем ако је текстуална спецификација написана на језику који није матерњи за дизајнера који треба да је преточи у одговарајућу имплементацију. Он је неразумљив и подложан је различитим интерпретацијама.

#### Опис помоћу таблица прелаза/излаза

Опис помоћу таблице прелаза/излаза је много формалнији начин спецификације жељеног понашања машине стања који оставља далеко мање простора за различите интерпретације од стране дизајнера. Овај опис обично се састоји из једне табеле у којој се за свако од могућих стања у којима се машина може наћи јасно наводе сви дозвољени прелази у наредно стање, као и потребне вредности излазних сигнала у том тренутку. Пример таблице прелаза/излаза приказана је на слици 2.1.3.2.1.

Table

Description automatically generated

Слика 2.1.3.2.1 – Таблица прелаза/излаза машине стања

На основу слике 2.1.3.2.1 можемо закључити да:

* Машина стања коју је потребно пројектовати има четири стања(S1, S2, S3, S4).
* Машина стања има два једнобитна улаза, x и y.
* Машина стања има један једнобитни Милијев излаз

Иако је таблични начин спецификације далеко бољи од текстуалног, ипак није довољно разумљив и тешко се на основу њега може „схватити логика рада“ машине стања коју је потребно пројектовати. Такође, било какве накнадне модификације машине морају се извршити врло пажљиво, јер је врло лако у таблици модификовати погрешно поље или заборавити да је потребно модификовати још неко додатно поље како би се променио начин рада машине у складу са нашим потребама.

#### Опис помоћу дијаграма стања

Опис помоћу дијаграма стања представља графички начин спецификације жељеног рада машине стања. Овај опис на компактан начин групише све неопходне информације за потпуни опис жељеног начина рада машине, а обзиром да је реч о графичком приказу, опис је далеко разумљивији за дизајнера од претходна два типа.

Дијаграм стања се састоји од скупа чворова, представљених помоћу овалних симбола, који су међусобно повезани усмереним гранама. Генеричка структура чвора који се користи приликом развоја дијаграма стања приказана је на слици 2.1.3.3.1.

Diagram

Description automatically generated

Слика 2.1.3.3.1 – Генеричка структура чвора унутар дијаграма стања машине

Сваки чвор у дијаграму стања представља једно, јединствено, стање у коме се може наћи машина стања. Сваки чвор мора имати јединствено симболичко име које му је придружено. Чвор на слици 2.1.3.3.1 има придружено симболичко име S0, које уједно представља и идентификацију стања у коме се машина налази у том тренутку.

Из сваког чвора може кретати једна или више излазних грана. Свака излазна грана представља једну могућу транзицију из текућег стања у неко друго стање. Поред сваке гране може, а и не мора, стајати логички израз који дефинише услов под којим се врши транзиција. Логички израз који специфицира услов у себи може да укључује улазне портове машине стања. Уколико из стања полази више од једне излазне гране, логички услови који су им придружени морају бити међусобно искључиви. У сваком тренутку мора бити задовољен само један од могућих услова, јер аутомат може прећи у само једно, наредно стање. Уколико из стања полази тачно једна грана, она мора бити безусловна, поред ње се не може налазити услов под којим ће се ова транзиција обавити. Ово је стога што у сваком тренутку мора бити једнозначно одређено наредно стање у коме ће се машина наћи.

Вредности излазних портова такође су специфициране на дијаграму стања. Нове вредности Мурових излаза наводе се унутар ознаке чвора (mo <= vrednost на слици 2.1.3.3.1), обзиром да су они само функција текућег стања у коме се аутомат налази. Обзиром да вредности Милијевих излаза зависе не само од текућег стања у коме се аутомат налази већ и од текућих вредности улазних портова аутомата, нове вредности Милијевих излаза смештају се поред сваке од излазних грана (me <= vrednost на слици 2.1.3.3.1).

Да би се поједноставио дијаграм стања, конвенција је да се на њему наводе доделе излазним портовима који у датом стању треба да се активирају. У случају да се излазни порт у датом стању не активира, он се не наводи на дијаграму, а подразумева се да излазни порт узима подразумевану вредност. Уобичајено је да се за подразумевану вредност излазних портова узима вредност 0, уколико то другачије није наглашено.

Diagram

Description automatically generated

Слика 2.1.3.3.2 – Дијаграм машине стања хипотетичког меморијског контролера

## Самоуслужни апарати (апарати за кафу)

Термин ”вендинг” апарати широј популацији и није толико познат, јер их већина једноставно зна под именом аутомати или апарати за самоуслуживање. Најчешћа места на којима могу да се виде су аутобуске станице, факултети, школе, канцеларије, хотели, болнице и слична места.

Аутомате можемо поделити у две врсте, а то су апарати за самопослуживање топлим, односно хладним напицима и ”грицкалицама”. Највећа предност вендинг апарата (самоуслужних апарата ) јесте то што потрошач производ купује већом брзином него у продавници и без чекања у реду. Понуда кафе, напитака и грицкалица је обимна, те купац увек има више избора када је неки производ у питању.

Самоуслужне апарате одликују практичност. Свако ко жели да без имало муке ужива у својој омиљеној кафи, или пак некој грицкалици ’’на ногама’’, има ту могућност. Иако постоје вековима уназад, самоуслужни апарати постају све модернији, познатији, али и траженији јер нуде потрошачима баш оно што им је потребно. Дају могућност да радници, ђаци, студенти, без напуштања установе, купе оно што им је неопходно.



Слика 2.2.1 – Самоуслужни апарат за кафу

### Историјат самоуслужних апарата

Прва забележена препорука за самоуслужне апарате налази се у раду Херона из Александрије, познатог инжењера и математичара из првог века. Његова машина је прихватала метални новчић, а затим издавала фиксну количину освештане воде. Кад се новчић депонује, пада у посуду везану за полугу. Полуга отвара вентил који омогућава води да исцури. Новчић наставља да се котрља вођен својом тежином све док не падне, након чега контра тег на полузи враћа полугу у пређашњи положај и затвара вентил.

Упркос овом раном преседану, аутомати су морали да чекају развој индустријског доба пре него што су постали познати. Прве модерне аутомате, машине које раде на металне новчиће су представљене у Лондону, у раним 80-им, и служиле су за издавање разгледница. Први овакав самостални аутомат у САД саграђен је 1888. године од стране компаније за израду жвакаћих гума “Tomas Adams”. Идеја додавања једноставних играчака у ове машине, као даљи подстицај ка анимацији купаца дошла је 1897, када је фирма Pulver додала малу фигуру, која би се кретала кад год би неко купио неку од жвакаћих гума из њихове машине. Ова једноставна идеја изродила је читав нови тип механичких уређаја познатих као уређаји за “анимирање купаца”. Настанком слот машина и флипера коначно је покренута производња и интерес за ове ране уређаје.

У децембру 1970. године на конвенцији у Даласу – Тексас, компанија ”Ussery industries”, приказала је свој нови "говорни" аутомат за продају, под називом Venda говорни аутомат. Убацивањем новчића, машина би изговарала: "Хвала Вам", уз глас комичара Henni Ioungmana. Venda је премијерно приказана у шоу програму Џонија Карсона у марту 1971.

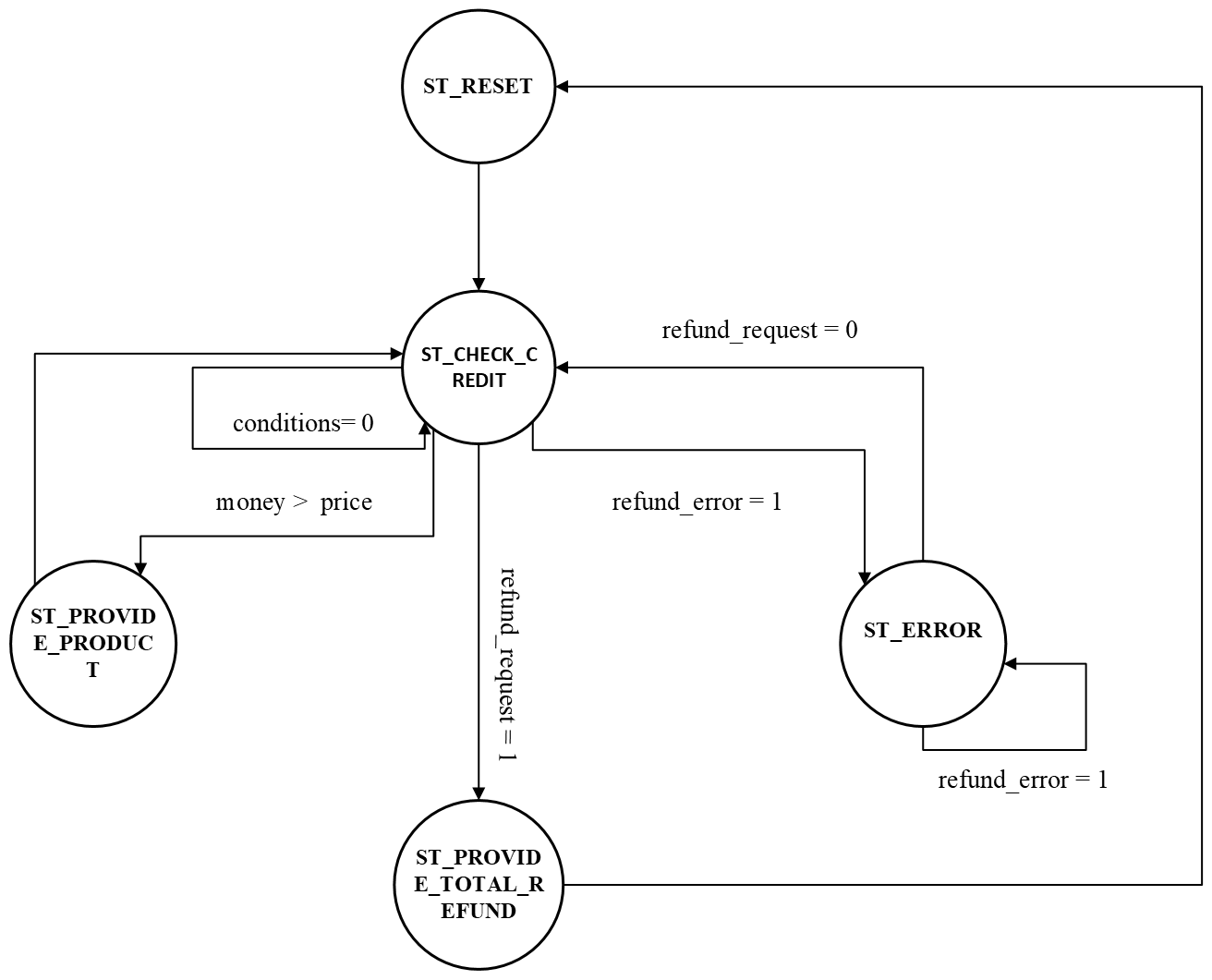
Самоуслужни аутомати су прошли кроз значајне промене током деценија. Данас многе машине имају могућност примања кредитне картице, магнетног кључа или жетона, а компаније које раде са самоуслужним апаратима имају могућност праћења стања апарата са даљине путем GPRS система. Апарати су добили нови и модернији облик и дизајн. Проширено је поље њихове намене. Тако да данас постоје самоуслужни апарати за кафу, сокове, воде, слатке и слане грицкалице, цигарете, новине, пошту, одећу, обућу, путничке карте...



## Дијаграм машине стања самоуслужног апарата за кафу

Самоуслужни апарат за кафу описан је дијаграмом машине стања приказан на слици 2.3.1. На слици 2.3.1 можемо приметити да постоји пет стања (ST\_RESET, ST\_CHECK\_CREDIT, ST\_PROVIDE\_ PRODUCT, ST\_PROVIDE\_TOTAL\_REFUND и ST\_ERROR) која омогућавају рад самоуслужног апарата за кафу. Када се апарат покрене тренутно стање (стање у коме се налазимо у датом тренутку) је ST\_RESET у коме се постављају иницијалне вредности одговарајућих сигнала. Након чега безусловно прелазимо у стање ST\_CHECK\_CREDIT у коме се проверава унети новац. У овом стању проверавамо вредност унетог новца и да ли купац има довољно новца да наручи напитак, такође проверавамо могућност прављења напитка и могућност повратка кусура . Стање ST\_CHECK\_CREDIT је стање у које се често враћамо како би омогућили накнадно додавање новца, поновне изборе, враћање кусура или уколико нема довољно новца у самоуслужном аутомату за повраћај кусура, враћа се новац који купац унесе. У овом стању смо све док су услови на нули. Услове које посматрамо су :

* Недостатак састојака за прављење пића (млеко, кафа, вода, чај, чоколада, шећер).
* Одабир пића за које нема довољно састојака.
* Укупно унети новац, као и да ли купац има довољно унетог новца за наручивање одабраног производа.
* Недостатак новца у самоуслужном апарату.
* Одабир производа.
* Одабир повраћаја новца.



Слика 2.3.1 – Дијаграм машине стања самоуслужног апарата за кафу

Уколико се неки од услова промени и са нуле пређе на јединицу, прелази се у следеће стање. На почетку се сматра да постоји довољно новца за враћање кусура, као и материјала потребних за прављење напитка. У зависности који је услов промењен прелазимо у стање ST\_PROVIDE\_ PRODUCT, ST\_PROVIDE\_TOTAL\_REFUND или ST\_ERROR.

Уколико купац има довољно новца (money > price) да наручи пиће прелази се у стање ST\_PROVIDE\_ PRODUCT. У овом стању се испоручује напитак након његовог прављења и враћа се на стање ST\_CHECK\_CREDIT, где купац бира да ли ће поново наручивати напитак или захтевати повраћај новца. Ако купац захтева повраћај новца прелази се у стање ST\_PROVIDE\_TOTAL\_REFUND. Стање ST\_PROVIDE\_TOTAL\_REFUND омогућава повраћај новца, где се након обављене радње враћа у прво стање ST\_RESET.

У стање ST\_ERROR прелази се из стања ST\_CHECK\_CREDIT само ако у самоуслужном апарату нема довољно новца за кусур (refund\_error = 1). Овде се враћа унети новац и онемогућава се избор било каквог напитка. Све док је refund\_error = 1 остаје се у стању ST\_ERROR, уколико се промени вредност refund\_error и постане нула прелази се у стање ST\_CHECK\_CREDIT.

# ОПИС МАШИНЕ СТАЊА HDL ЈЕЗИКОМ

У данашње време се велики број дигиталних система имплементира у програмабилним логичким (PLD, енгл. *Programmable Logic Device*) из два разлога. Први разлог коришћења је широка примена у пракси, као и њихова погодност за пројектовање различитих типова дигиталних кола. Други разлог се огледа у начину имплементације дизајна у овим колима, која се постиже помоћу програмирања од стране корисника.

Имамо три главна типа PLD кола:

* SPLD (енгл. *Simple PLD*)
* CPLD (енгл. *Complex PLD*)
* FPGA (енгл. Field Programmable Gate array)

За унос пројеката у CAD систем, превод пројеката у одабрану компоненту, симулирање функционалности и временска зависност резултујућег кола, имплементацију пројеката у компоненте, пројектанти користе језик за опис хардвера (HDL,енгл*. Hardware Description Language*).

Језици за опис хардвера личе на програмске језике, али су специјално оријентисани на описивање структуре хардвера и понашања. Коришћењем HDL језика се обезбеђује алтернатива за шематски приказ кола. Уколико се HDL језик користи као алтернатива шематском приказу кола, он је најчешће структурног описа и као на шеми, он повезује компоненте. Такав структурни опис познат је под називом „netlist“.

Постоје два HDL језика која су широко коришћени стандардни језици за опис хардвера :

* VHDL (енгл. *Very high speed integrated circuit HDL*)
* Verilog HDL

Они су званично одабрани и стандардизовани од стране IEEE-а (енгл. *Institute of Electrical and Electronics Engineers*). Поред стандардних језика бројне фирме имају своје интерне језике који се могу користити само за реализацију кола користећи технологију фирме којој језик припада. Као IEEE стандарде, VHDL и Verilog HDL подржавају сви произвођачи дигиталног хардвера.

Још један битан HDL језик је AHDL (енгл. *Altera HDL*) развијен од стране компаније Алтера. Омогућава погодан начин за конфигурацију логичких кола које Алтера нуди, користи се за програмирање комплексних дигиталних система у Алтера програмабилном логичком колу.

## VHDL

VHDL је језик који служи за описивање функционисања хардвера, као и за дизајн дигиталних електронских система, попут основних дигиталних кола, кориснички програмабилних логичких низова (енгл. *Field Programmable Gate Array -* FPGA) и кола са специјалном наменом (енгл. *Application-Specific Integrated Circuit* - ASIC) VHDL језик може да се користи у свим фазама пројектовања интегрисаних кола: опису, симулацији и синтези интегрисаних кола. Систем који се развија у VHDL-у, може се физички имплементирати на FPGA, ASIC или CPLD чиповима. VHDL је скраћеница енглеских речи *VHSIC Hardware Description Language*, где се под *VHSIC* (енгл. *Very High Speed Integrated Circuit*) сматра интегрисано коло врло велике брзине. Сложеност дигиталних система који се моделују може варирати од једноставног логичког кола па све до комплетног дигиталног система. Осим VHDL-а, за опис понашања дигиталног електронског кола (система) могу се користити и ABEL, AHDL, Verilog и слично.

VHDL је у суштини паралелан језик, а не само структуралан као што је рецимо Pascal или објектно оријентисан као што је C++. Под појмом паралелан се подразумева да се елементи VHDL програма извршавају паралелно (истовремено), за разлику од структуралних програмских језика где се програм извршава по принципу “наредба по наредба”. Наредбе у оквиру процеса, функција и процедура у VHDL-у се извршавају секвенцијално.

### Структура VHDL фајла

Основни делови сваког фајла који садржи VHDL код којим се описује неко дигитално коло, састоји се од ентитета и архитектуре. Ентитет описује основне улазе и излазе у/из кола, док архитектура описује функционалност датог кола, тј. претварање датих улаза у одговарајуће излазе. Поред ова два дела, постоји и подручје за декларисање библиотека, које садрже скуп често коришћених функција. Уколико се нека функција смести у библиотеку, омогућено је њено поновно коришћење непосредним позивањем функције. Општа структура VHDL фајла приказана је на слици 3.1.1.1.

A picture containing graphical user interface

Description automatically generated

Слика 3.1.1.1 – Општа структура VHDL фајла

#### Деклерација библиотека

Библиотеке у VHDL-у представљају скупове наредби, функција, дефинисаних типова података и операција које омогућавају једноставнију употребу језика за пројектовање дигиталних кола и система.

За декларацију библиотека потребне су две линије кода: прва која служи за назив библиотеке, а друга која помоћу USE наредбе омогућава коришћење садржаја наведене библиотеке. Општа структура за позивање и укључивање неке библиотеке дата је сљедећим кодом приказаним на слици 3.1.1.1.1.

Text

Description automatically generatedСлика 3.1.1.1.1 – Општа структура за укључивање библиотеке

Већина библиотека састоји се из великог броја пакета, тако да је потребно додатно специфицирати који пакет или који део пакета се користи.

Три најчешће коришћене библиотеке су:

* ieee
* standard
* work

Библиотеке **standard** и **work** су аутоматски укључене у сваком VHDL пројекту, тако да се не морају експлицитно назначити. Библиотека **ieee** се користи само онда када се користи *STD\_LOGIC* врста података.

#### Дефинисање ентитета

Ентитет представља спољашњи део кола које се пројектује, односно дефинисање ентитета служи за специфицирање улазних и излазних сигнала кола које се пројектује. Општи облик декларације ентитета приказан је на слици 3.1.1.2.1.

Text

Description automatically generated

Слика 3.1.1.2.1 – Општи облик деклерације ентитета

**ENTITY** означава да је започето декларисање ентитета, затим следи назив ентитета (који корисник сам задаје), док реч **IS** означава почетак дефинисања улазних и излазних сигнала за дати ентитет. Помоћу речи **PORT** дефинишу се улазни и излазни портови сигнала тако што се најпре зада име сигнала (које може било које валидно VHDL име, односно идентификатор), затим се задаје режим рада порта и на крају тип податка које тај порт користи.

Режим рада порта одређује његову улогу у функционисању кола и система који се пројектује. У зависности од намене режим рада порта може бити:

* Улазни
* Излазни
* Улазно-излазни
* Бафер

#### Дефинисање архитектура

Архитектура кола описује начин рада и дефинише функцију самог кола, односно врши функционално повезивање улаза и излаза кола. Општа структура дефинисања архитектуре дата је на слици 3.1.1.3.1.

Graphical user interface, text, application

Description automatically generated

Слика 3.1.1.3.1 – Општа структура дефинисања архитектуре

**ARCHITECTURE** најпре се задаје име архитектуре за одговарајући ентитет (OF ime\_entiteta), а затим се речју **IS** опционо врше декларације. **BEGIN** почиње дефинисање тела архитектуре, а дефинисање архитектуре завршава се речју **END** након које следи назив архитектуре.

Дефинисање архитектуре се састоји из два главна дела:

* Област декларације
* Тело архитектуре

Област декларације може се опционо користити и њено постојање није обавезно при дефинисању архитектуре. Област декларације јавља се пре речи **BEGIN** и може се користити за декларисање помоћних сигнала (SIGNAL), константи (CONSTANT), кориснички дефинисаних типова података (TYPE), компоненти (COMPONENT) и атрибута (ATTRIBUTE).

**BEGIN** означава почетак тела архитектуре, односно функционалног дела самог кола. У оквиру тела архитектуре дефинишу се искази на основу којих се извршавају одређене логичке или аритметичке операције

### Основни елементи VHDL језика

Основни елементи VHDL језика су:

* Идентификатори
* Објекти података
* Типови података
* Оператори

#### Идентификатори

У VHDL језику идентификатори се састоје од низа једног или више карактера. У дозвољене карактере спадају:

* Велика и мала слова (од А до Z)
* Бројеви (од 0 до 9)
* Повлака („\_“)

Први карактер мора да буде слово, а задњи карактер не може бити повлака, такође VHDL језик не разликује мала и велика слова.

Коментар се у VHDL језику наводи са две узастопне црте („--“) на крају линије.

#### Објекти података

Постоје три врсте података које се могу дефинисати у VHDL језику:

* Константе – именују одређене вредности
* Променљиве – користе се за локално смештање привремених података
* Сигнали – повезују портове/пинове компоненти

Константе садрже само једну вредност датог типа података и не могу се мењати током извршавања кода. Константе се употребљавају у програму или моделу када се једна вредност користи више пута, на овај начин мењање вредности је доста лакше.

Деклерација константе приказана је на слици 3.1.2.2.1.

A picture containing text

Description automatically generatedСлика 3.1.2.2.1 – Деклерација константе у VHDL језику

Деклерација променљиве је приказана на слици 3.1.2.2.2.

Text

Description automatically generated with low confidenceСлика 3.1.2.2.2 – Деклерација променљиве у VHDL језику

Након речи **VARIABLE** могуће је дефинисати више променљивих раздвојених зарезом. Врста променљиве, као што и име каже, дефинише врсту вредности дефинисану променљивом.

Деклерација сигнала дата је на слици 3.1.2.2.3.

A black text on a white background

Description automatically generated with medium confidence

Слика 3.1.2.2.2 – Деклерација сигнала у VHDL језику

Након речи **SIGNAL** може се дефинисати више сигнала раздвојених зарезом. Врста сигнала дефинише тип података који се користи за наредни сигнал.

Сигнали се могу дефинисати приликом дефинисања ентитета, архитектуре и пакета. Уколико се вредност сигнала промени у току процеса, сигнал ће ту вредност попримити тек по завршетку процеса, што значи да се у току тренутног процеса не може користити, за разлику од променљивих коју дату вредност тренутно поприма.

#### Типови података

Да би написали VHDL код потребно је знати који су дозвољени типови података, како их специфицирати и користити. Сваки објекат података у VHDL-у садржи вредност која припада неком низу вредности, а тај низ је специфициран коришћењем декларације типа. Одређени типови и операције које се могу спровести над објектом су дефинисани у самом језику. Декларација унапред дефинисаних типова података налази се у пакету STANDARD и оператори дозвољени над овим типовима су дефинисани у самом језику. Декларација унапред дефинисаних типова података налази се у библиотекама попут библиотеке **std** у пакету стандард која дефинишу типове података: **BIT**, **BOOLEAN**, **INTEGER** и **REAL**, библиотеке **ieee** у пакету **std\_logic\_1164**, која дефинише типове **STD\_LOGIC** и **STD\_LOGIC\_VECTOR**.

**STD\_LOGIC** има следеће могуће вредности:

* U ( Uninitialized ) - ако нема специфичне почетне вредности
* X (Forcing Unknown ) - 'X' јавља се када се не може донети одлука да ли је “0” или ”1”
* 0 (Forcing 0)
* 1 (Forcing 1)
* Z (High Impedance)
* W (Weak Unknown)
* L (Weak 0)
* H (Weak 1)
* - ( Don't care)

У VHDL језику постоји могућност дефинисања новог типа података, као и увођење ограничења за постојеће типове података.

#### Оператори

У VHDL језику постоје неколико предефинисаних оператора:

* Оператори доделе
* Логички оператори
* Аритметички оператори
* Релациони оператори
* Оператори помераја бита
* Оператори спајања

Оператори доделе се користе за додељивање вредности сигнала, променљивих и константи („ <= “- додељивање вредности сигнала , „ := “- додељивање вредности променљивим и константама, „ => “- додељивање вредности појединачним елементима вектора).

Логички оператори се користе над подацима типа BIT и STD\_LOGIC, као и над њиховим продужецима, BIT\_VECTOR и STD\_LOGIC\_VECTOR (NOT, AND, OR, NAND, NOR, XOR, XNOR).

Аритметички оператори се користе над подацима типа INTEGER, SIGNED, UNSIGNED или REAL („+“ - сабирање, „-“ - одузимање, „\*“ - множење, „/“ - дељење, „\*\*“ - експонент, „MOD“ – модул броја , „REM“ – остатак при дељењу, „ABS“ – апсолутна вредност).

Релациони оператори се користе над подацима типа INTEGER, SIGNED, UNSIGNED или REAL („ = “ – једнако, „ /= “ – није једнако, „ < “ – мање од, „ > “ – веће од, „ <= “- мање или једнако, „ >= “ – веће или једнако). Они се начеше користе код условних наредби.

## Имплементација у VHDL језику

За опис машине стања самоуслужног апарата коришћен је VHDL језик. На сликама 3.2.1, 3.2.2, 3.2.3, 3.2.4 , су приказани делови кода који ће у наставку текста поступно бити објашњени.

Text

Description automatically generated with medium confidence

Слика 3.2.1 – Опис самоуслужног апарата у VHDL језику (I-део)

A picture containing graphical user interface

Description automatically generated

Слика 3.2.2 – Опис самоуслужног апарата у VHDL језику (II део)

Text

Description automatically generated

Слика 3.2.3 – Опис самоуслужног апарата у VHDL језику (III део)

Graphical user interface, text, application

Description automatically generated

Слика 3.2.4 – Опис самоуслужног апарата у VHDL језику (IV део)

Text

Description automatically generated with medium confidence

Слика 3.2.5 – Дефинисање улазних и излазних сигнала

Назив entity-а мора да буде исти као и назив фајла, у овом случају назив фајла је Aparat\_za\_kafu. У њему смо дефинисали улазне и излазне сигнале потребне за рад самоуслужног апарата. На слици 3.2.5 приказани су улазни и излазни сигнали.

Улазни дефинисани сигнали су:

* i\_refound – говори нам да ли има довољно новца у апарату за повраћај кусура
* i\_clk – импулсни сигнал
* i\_rstb – сигнал за рестарт
* i\_refund\_request – захтевање повраћаја кусура
* i\_milk – говори нам да ли има млека у апарату
* i\_chocolate – говори нам да ли има чоколаде у апарату
* i\_sugar - количина шећера
* i\_sugar\_valid – говори нам да ли има шећера у апарату
* i\_coffe – говори нам да ли има кафе у апарату
* i\_water – говори нам да ли има воде у апарату
* i\_tea – говори нам да ли има чаја у апарату
* i\_product\_sel – избор производа
* i\_money\_value – количина унетог новца
* i\_money\_value\_valid – показатељ да ли је унети новац валидан

Излазни дефинисани сигнали су :

* o\_product\_delivery\_sel – производ који се продао (нпр. Капућино, чај, топла чоколада и сл. )
* o\_product\_delivery\_sel\_ena – испорука изабраног производа (пиће је направљено и испоручено)
* o\_money\_refund\_value – износ кусура
* o\_money\_refund\_value\_ena – потврда да је кусур испоручен

Неки од улазних и излазних сигнала нису једнобитни, као што може да се примети на слици 3.2.5, они су више битни и дефинишу се командом std\_logic\_vector(1 downto 0). Број у загради зависи од броја бита који нам је потребан .

### Архитектура

У Architecture се дефинишу процеси, стања, додатни сигнали и сл. На почетку се уводе константе које служе да се напицима доделе цене, у овом случају напици могу да коштају четрдесет, педесет или шездесет динара. Поред дефинисаних константи на слици 3.2.1.1 се може видети и начин прављења новог типа за бројни низ који служи да рачуна унети новац купца, чиме га обавезујемо да може да унесе вредности задате у овом низу.

Text

Description automatically generated

Слика 3.2.1.1 – Декларисање константи и прављење новог типа за низ

Text

Description automatically generated

Слика 3.2.1.2 – Дефинисање стања и деклерација сигнала

На слици 3.2.1.2. можемо приметити да имамо пет стања (ST\_RESET, ST\_CHECK\_CREDIT, ST\_PROVIDE\_PRODUCT, ST\_PROVIDE\_TOTAL\_REFUND, ST\_ERROR). Ова стања нам говоре где се налазимо у датом тренутку, такође поштовањем шеме машине стања (слика 2.3.1 на страни 12.) и увођењем услова пребациваћемо се из једног стања у друго. На страни дванаест, а и касније у тексту биће више о сврси и условима преласка стања.

Сигнали (слика 3.2.1.2) r\_price\_product, r\_price\_product\_counter и r\_refund су бројног типа и везани су за цену изабраног производа, укупну количину новца и кусур, док r\_st\_present и w\_st\_next су сигнали везани за тренутно и следеће стање. r\_st\_present узима вредност тренутног стања и оно се не мења док се наше тренутно стање не промени , а w\_st\_next има вредност следећег стања.

Graphical user interface, text, application, email

Description automatically generated

Слика 3.2.1.3 – Дефинисање стања и деклерација сигнала

На слици 3.2.1.3 приказана су три процеса (p\_state, p\_comb и p\_state\_out) и конверзија сигнала у unsigned вредност ( линија 170 на слици 3.2.1.3).

У први процес p\_state (слика 3.2.1.4) улазимо уколико сигнали i\_clk или i\_rstb промене вредност, а пошто се вредност сигнала i\_clk стално мења у овај процес се увек улази. Уколико је i\_rstb једнак нули тренутно стање у коме се налазимо је ST\_RESET, а у супротном на сваку узлазну ивицу i\_clk сигнала тренутно стање постаје следеће стање (r\_st\_present <= w\_st\_next).

Text

Description automatically generated

Слика 3.2.1.4 – Први процес p\_state

Да би се ушло у процес, неки од сигнала наведених у загради мора да промени вредност. То важи за све наведене процесе и уласке у њих, где им је једина разлика сигнал који мора бити промењен (услов). На слици 3.2.1.5 приказан је други процес p\_comb и његови условни сигнали чија промена вредности омогућава улаз у процес.

Graphical user interface, text, application

Description automatically generated

Слика 3.2.1.5 – Услови(сигнали) за улазак у други процес

У другом процесу користи се наредба case (слика 3.2.1.6), којом се описује у које стање се прелази под којим условима и које је следеће стање.

Text

Description automatically generated

Слика 3.2.1.6 – Одређивање тренутног и следећег стања, други процес p\_comb, I

Када је r\_st\_present (слика 3.2.1.6) ST\_CHECK\_CREDIT (проверава се новац и бира пиће) следеће стање w\_st\_next је ST\_ERROR (грешка, немогуће је наручити било које пиће) уколико је i\_refound (у апарату нема довољно новца за повратак кусура) једнак јединици (линије 71,72,73 слика 3.2.1.6). У супротном ако је (линије 74 и 75 слика 3.2.1.6) i\_refund\_request (захтевање повраћаја кусура) једнак јединици и r\_price\_product\_counter (укупно пара колико се налази у апарату пре и после наручивања напитка) различито од нуле, следеће стање (w\_st\_next) ће бити ST\_PROVIDE\_TOTAL\_REFUND (враћа се кусур купцу). Линије 76, 79, 81, 83, 86 су услови везани за недостатак материјала у апарату и они онемогућавају прелазак у следеће стање, ако се изабере пиће које садржи ове материјале. Ако покушамо на пример да изаберемо капућино са чоколадом, а у апарату нема више чоколаде, радња неће бити прихваћена и купац ће морати да изабере друго пиће уколико жели или може да узме новац назад. Што се тиче линије 88 која је везана за шећер, десиће се иста ствар, стање се неће мењати док купац не изрази жељу да хоће пиће без шећера или натраг новац. Такође купцу је онемогућено да изабере производ који није на листи, линија 90 на слици 3.2.1.7 (уколико се на апарату налази више тастера, а нема довољно врста пића).

Text

Description automatically generated

Слика 3.2.1.7 – Одређивање тренутног и следећег стања, други процес p\_comb II

У супротном проверава се да ли је унета количина новца довољна за куповину изабраног пића, па ако јесте прелази се у стање ST\_PROVIDE\_PRODUCT (линије 93, 94, 95 слика 3.2.1.7), а уколико није остаје се у стању ST\_CHECK\_CREDIT где може да се дода новац.

Graphical user interface, text, application

Description automatically generated

Слика 3.2.1.8 – Одређивање тренутног и следећег стања, други процес p\_comb III

Када је тренутно стање ST\_PROVIDE\_PRODUCT безусловно се прелази у следеће стање ST\_CHECK\_CREDIT (линије 101, 102 слика 3.2.1.8), где имамо могућност након завршене куповине и узимања првог напитка да изаберемо да ли ћемо узети назад новац или поново наручити пиће. Из стања ST\_RESET безусловно се прелази у стање ST\_CHECK\_CREDIT (линије 106, 107 слика 3.2.1.8). Када је аутомат у стању ST\_ERROR, ако је i\_refound једнако нули иде се у стање ST\_CHECK\_CREDIT. У супротном остаје се у стању ST\_ERROR, све док је i\_refound једнак јединици (линије 109, 110, 111, 112, 113 слика 3.2.1.8 ).

Трећем стању p\_state\_out (слика 3.2.1.9) приступа се уколико се вредност сигнала i\_clk или i\_rstb промени. Када се приступи овом стању, уколико је i\_rstb једнак нули (линије 119, 120, 121, 122, 123) враћају се четири сигнала на почетну вредност (r\_refund, o\_money\_refund\_value\_ena, o\_product\_delivery\_sel, o\_product\_delivery\_sel\_ena), тј. враћају се на нулу. У супротном се на узлазној ивици i\_clk сигнала улази у case.

Када је сигнал r\_st\_present стање ST\_CHECK\_CREDIT, o\_product\_delivery\_sel\_ena (производ је направљен и испоручен) добија вредност нула (линије 125, 126, 127 слика 3.2.1.9).

A picture containing text

Description automatically generated

Слика 3.2.1.9 – Треће стање p\_state\_out I

Уколико је унети новац валидан (i\_money\_value\_valid = '1'), онда уколико је i\_money\_value мање од дефинисаног ограничења за уношење новца, r\_price\_product\_counter се повећава за онолико пара колико унесемо (линије 128, 129, 130). Због раније дефинисаних новчаница које могу да се унесу и претходног враћања вредности на нулу, r\_price\_product\_counter ће сабирати вредност новца сваки пут када се унесе. На пример, смемо унети пет, десет, двадесет, педесет, сто или двеста динара и уколико желимо купити пиће у вредности од шездесет динара, унећемо новчанице од педесет и од десет динара. Уношење већих новчаница попут петсто, хиљаду, две хиљаде и пет хиљада неће бити прихваћено.

На слици 3.2.1.10 приказан је начин доделе цене напицима, под условом да изабрани напитак није нулти и није већи од једанаестог (ови напици не постоје, линија 135 слика 3.2.1.10). У колико је овај услов испуњен задајемо нове услове који нам разврставају пића и додељују им цену (линије 136, 137, 138, 139, 140, 141). На линији 143 задајемо вредност производу у износу од 41, међутим никада се не долази у ову линију и служи само да би симулација радила, јер бит за број један сматра не дефинисаним, јер се не користи, па се цене производа не исписују како треба.

Када је тренутно стање r\_st\_present у стању ST\_PROVIDE\_PRODUCT, онда одузимамо цену купљеног производа од укупног новца који смо убацили у апарат (линија 149 слика 3.2.1.11), сигнал o\_product\_delivery\_sel\_ena постаје један чиме се показује да је напитак успешно испоручен, o\_money\_refund\_value\_ena и r\_price\_product добијају вредност нула (линије 150, 151, 152 слика 3.2.1.11).

Када је тренутно стање r\_st\_present у стању ST\_PROVIDE\_TOTAL\_REFUND сигнал r\_refund добија вредност сигнала r\_price\_product\_counter, што је заправо вредност кусура који купац добија. Сигнал o\_product\_delivery\_sel\_ena добија вредност нула, чиме се говори да нема производа који се прави, који је прихваћен или испоручен и сигнал o\_money\_refund\_value\_ena добија вредност један, што показује да је купац затражио натраг новац и добио кусур (линије 154, 155, 156, 157 слика 3.2.1.11).

Када је тренутно стање r\_st\_present у стању ST\_RESET, враћају се сигнали на почетну вредност, односно на нулу (линије 159, 160, 161, 162, 163 слика 3.2.1.11).

Када је тренутно стање r\_st\_present у стању SR\_ERROR онда се враћа убачен новац (линије 164, 165, 166 слика 3.2.1.11).

Text

Description automatically generated

Слика 3.2.1.10 – Треће стање p\_state\_out II, додељивање цена

Text

Description automatically generated

Слика 3.2.1.11 – Треће стање p\_state\_out III

# ИМПЛЕМЕНТАЦИЈА ДИЗАЈНА У ПРОГРАМАБИЛНОМ ЛОГИЧКОМ КОЛУ И ТЕСТИРАЊЕ

За тестирање кода користи се „тест“ шема, реализована у графичком едитору, где су убачени сви улазни сигнали и од VHDL кода је направљен симбол. Такође, за бољу реализацију убачен је и детектор успонске ивице.

## Детектор успонске ивице и његова примена на тест шеми

Детектори ивица представљају кључне градивне блокове у дигиталним системима, омогућавајући екстракцију ивичних прелаза, као што су узлазне или силазне ивице, из дигиталних сигнала. Детектори узлазних ивица су посебно дизајнирани да детектују и реагују на узлазне ивице, које означавају прелазак из ниског у високи ниво напона у улазном сигналу. Могућност прецизног детектовања узлазних ивица кључна је за примене које укључују синхронизацију, покретање догађаја и прецизно временско усклађивање.

Детекција узлазних ивица подразумева хватање и анализу промена у улазном сигналу. Основни принципи се заснивају на коришћењу D флип-флопова, логичких капија и механизама за временско усклађивање. D флип-флопови служе као елементи за складиштење, забележавајући промене стања улазног сигнала. Логичке капије, попут XOR, OR и NOT капија, користе се за поређење тренутног и одложеног стања улазног сигнала и генерисање сигнала детекције узлазних ивица.

Постоје различите имплементације детектора узлазних ивица, при чему свака има своје предности и посебне карактеристике. Уобичајени приступи укључују коришћење D флип-флопова, XOR капија, AND капија, OR капија и NOT капија. Ове компоненте се међусобно повезују како би се забележиле промене стања, генерисали одложени сигнали, упоредили тренутно и одложено стање и произвео сигнал детекције узлазних ивица.

Детектори узлазних ивица налазе широку примену у разним областима, што показује њихову ширу применљивост и значај. У системима за комуникацију података, они омогућавају прецизну синхронизацију и обезбеђују тачан пренос података активирањем акција на основу узлазних ивица. Такође су кључни у обради сигнала, где детекција узлазних ивица омогућава прецизно бележење и анализу догађаја. Осим тога, детектори узлазних ивица се користе у системима за управљање, дигиталним временским колима и апликацијама у реалном времену како би се обезбедило прецизно покретање и синхронизација.

Упркос широкој употреби и поузданости, детектори узлазних ивица суочавају се са питањима интегритета сигнала и ограничењима времена. Напредак у технологији интегрисаних кола и методологијама дизајна довео је до развоја поузданијих и ефикаснијих техника детекције узлазних ивица. Ови напреци имају за циљ решавање изазова као што су имунитет на шум, потрошња енергије и брже деловање.

У овом случају детектор узлазне ивице користили смо како би омогућили да се новац региструје једанпут, уместо више пута, због брзине сигнала такта .

На следећој слици (слика 4.1.1) је приказана шема детектора узлазне ивице.

A picture containing diagram, plan, line, technical drawing

Description automatically generated

Слика 4.1.1 – Шематски приказ детектора узлазне ивице

Да би боље објаснили и проверили исправност рада детектора успонске ивице, направљена је симулација. На слици (слика 4.1.2) је приказана симулација рада детектора узлазне ивице.

A picture containing text, screenshot, diagram, line

Description automatically generated

Слика 4.1.2 – Симулација рада детектора узлазне ивице

На сваку узлазну ивицу улазног сигнала, добијамо такт на излазу, који нам омогућава касније, тачније бележење и регистровање унетог новца.

## Тест шема самоуслужног апарата

Од VHDL кода приказаног и објашњеног у поглављу 3.2 направљена је компонента (симбол) приказан на следећој слици (слика 4.2.1).

A screenshot of a computer program

Description automatically generated with low confidence

Слика 4.2.1 – Симбол апарат за кафу

На слици се виде улазни сигнали (i\_refound, i\_clk, i\_rstb, i\_refund\_request, i\_mikl, i\_chocolate, i\_sugar[1..0], i\_sugar\_valid, i\_coffe, i\_water, i\_tea, i\_product\_sell[3..0], i\_money\_value[2..0] и i\_money\_value\_valid), као и излазни сигнале (o\_product\_delivery\_sell[3..0], o\_product\_delivery\_sel\_ena, o\_money\_refund\_value[7...0] и o\_money\_refund\_value\_ena).

На тест шеми се поред симбола за апарат за кафу налази и детектор успонске ивице који обезбеђује да се убачен новац региструје једном. Детектор узлазне ивице има два улазна сигнала SLCK и i\_money\_value [2..0], а излаз иде на улаз i\_money\_value\_valid симбола апарата за кафу (слика 4.2.1).

A picture containing text, line, number, plot

Description automatically generated

Слика 4.2.1 – Тест шема са додељеним пиновима

## Симулација тест шеме самоуслужног апарата

Тестирањем у симулацији проверава се рад тест шеме као и тачност и исправност кода. За потребе симулације користе се улазни сигнали, регистри, стања и излазни сигнали, као што је приказано на слици 4.3.1.

A screenshot of a computer

Description automatically generated with medium confidence

Слика 4.3.1 – Увезени сигнали

Након тога треба подесити end time и grid size, тако што кликнемо на падајући мени где пише edit, након тога изаберемо опцију end time и наместимо на једну мили секунду (слика 4.3.2 и слика 4.3.3).

A screenshot of a computer

Description automatically generated

Слика 4.3.2 – Падајући мени edit и приказ две потребне опције end time и grid size

A screenshot of a computer

Description automatically generated with medium confidence

Слика 4.3.3 – End time, намештање на једну мили секунду

Након подешавања end time подешава се и grid size, тако што је периода стављена на десет микро секунди (слика 4.3.4).

A screenshot of a computer

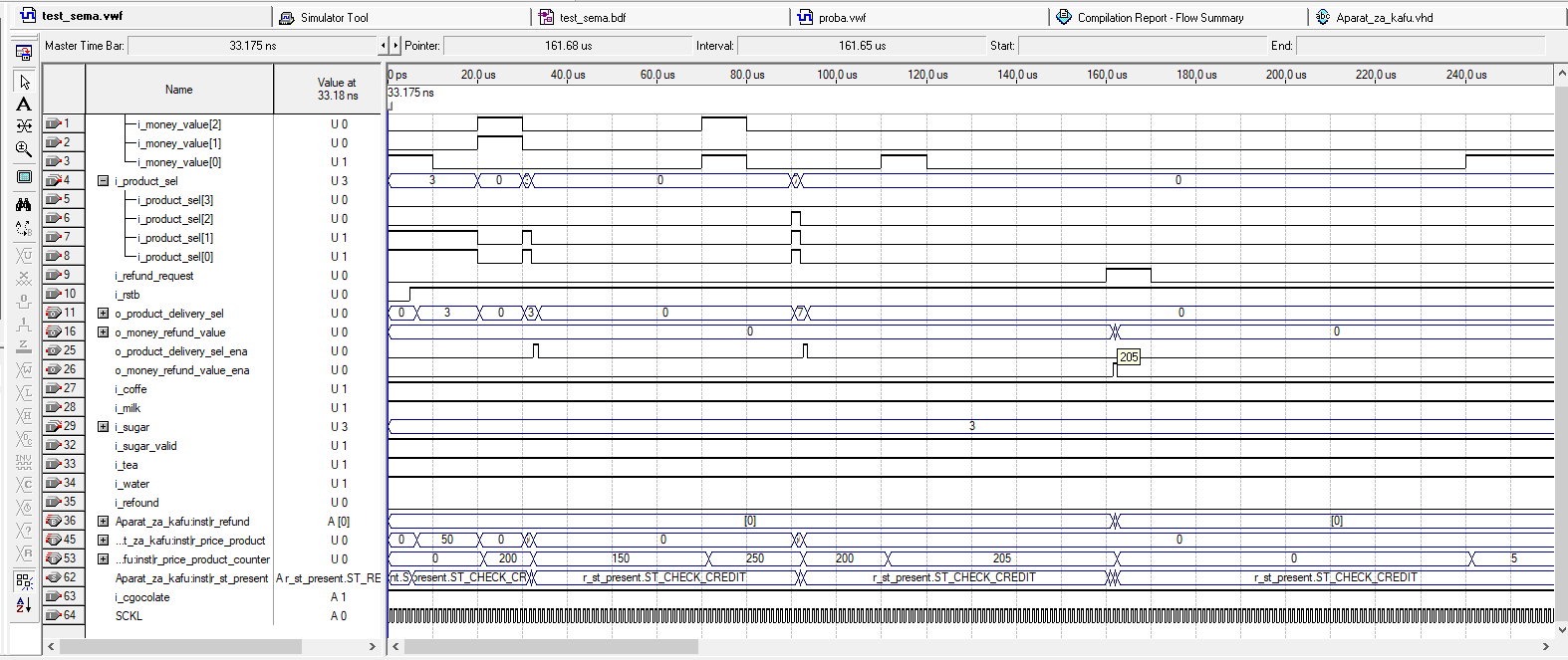
Description automatically generated with medium confidence

Слика 4.3.4 – Grid Size, намештање на десет микро секунди

На крају се подешавају вредности улазних сигнала. У овом случају је требало задати које се пиће наручује, чега ће све бити (да ли има млека, кафе, чоколаде, чаја, воде шећера и слично), колико новца је убачено, брзину сигнала такта, да ли у апарату има довољно новца за кусур, када је тражен повраћај новца и слично.

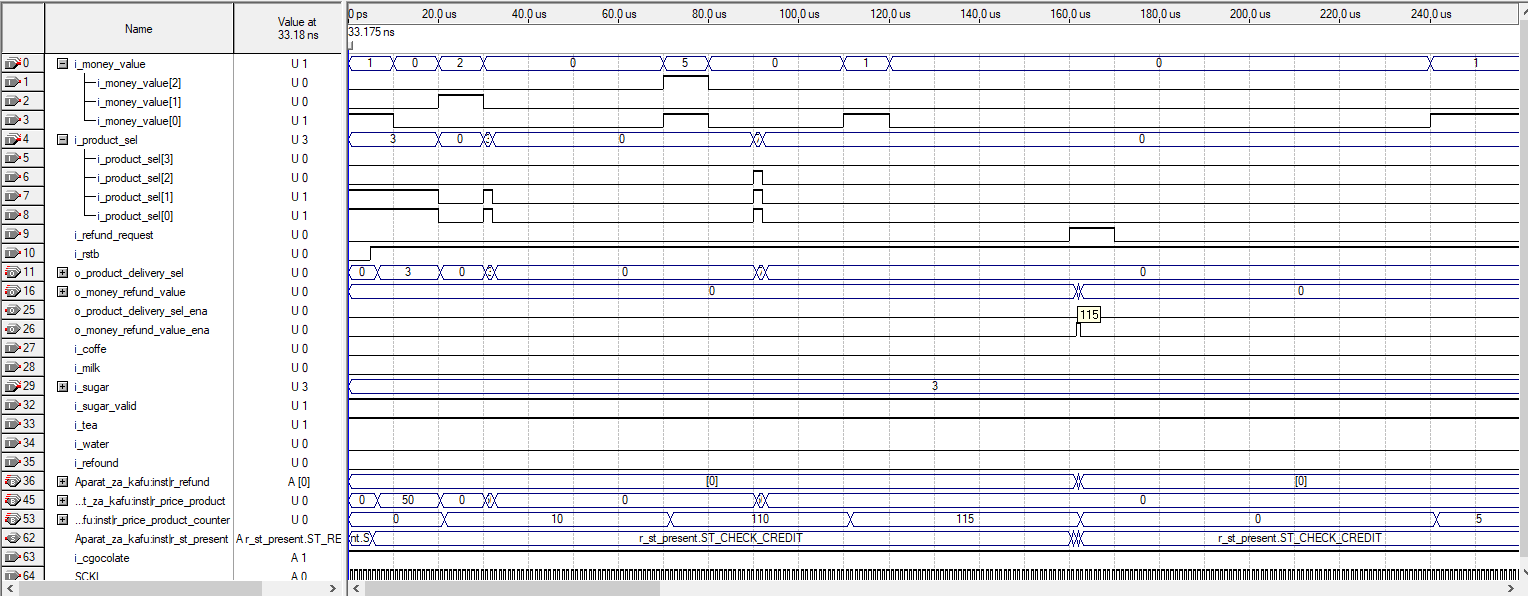
Овим намештањем издвојили смо следеће три могуће ситуације:

1. У првом случају апарат има довољно новца за кусур, као и сав потребан материјал за прављење било ког пића (слика 4.3.5)
2. У другом случају апарат нема сав потребан материјал и не може да направи одабрано пиће (слика 4.3.6)
3. У трећем случају, апарат нема довољно новца да врати кусур и одбија поруџбину и јавља се грешка (слика 4.3.7)



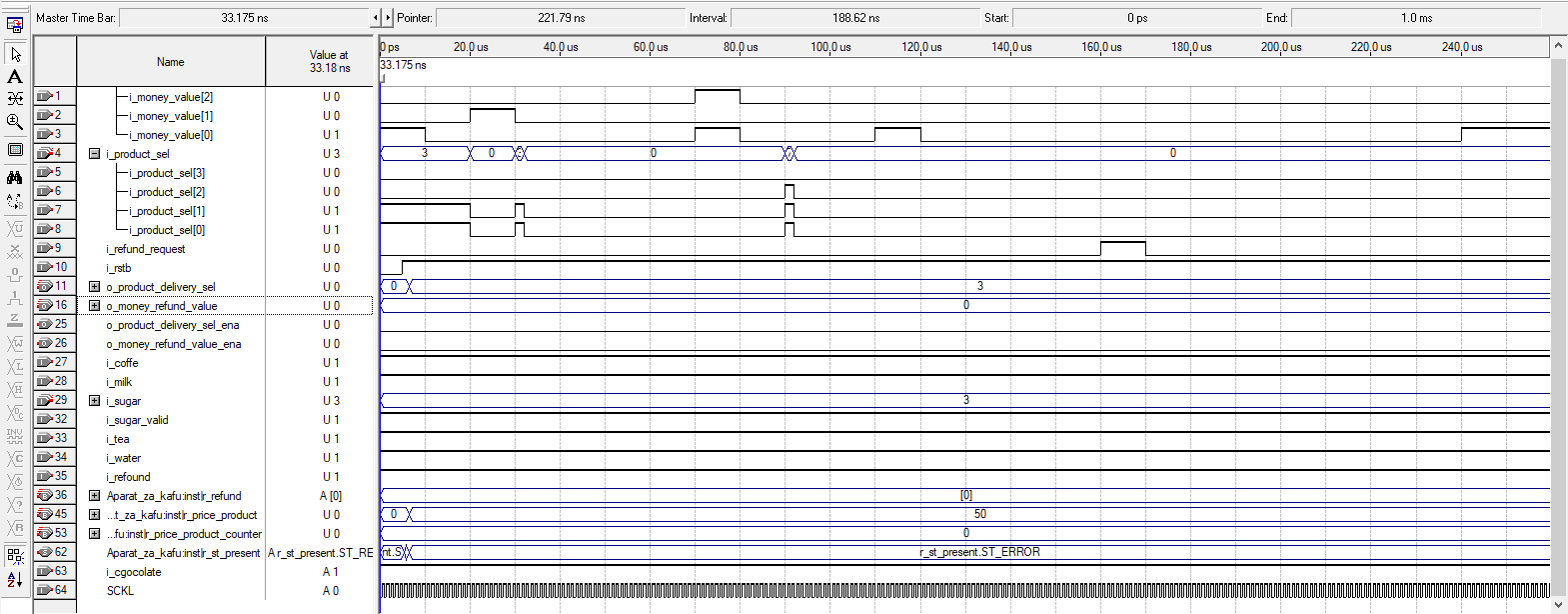
Слика 4.3.5 – Први случај - апарат нормално ради и враћа кусур и прави одабрано пиће

У симулацији за други случај, показано је како се пиће не прави и корисник има избор између поновног наручивања и повраћаја новца. На слици 4.3.6 је приказано да корисник захтева повраћај новца, али битно је напоменути да има и могућност и избора другог пића.



Слика 4.3.6 – Други случај - нема потребних материјала за прављење пића и апарат враћа новац

Трећи случај, приказан на слици 4.3.7 је ситуација грешке, у коме је спречено да се особи не врати кусур, тако што апарат враћа унети новац и улази у стање грешке.



Слика 4.3.7 – Трећи случај - нема довољно новца за кусур, апарат одбија поруџбину, враћа унети новац и улази у стање грешке

# ЗАКЉУЧАК

У завршном раду описана је машина стања, којом је описан рад самоуслужног апарата за кафу, применом VHDL језика и извршена је симулација рада предложеног решења самоуслужног апарата.

У предложеном решењу, решена су и два проблема која могу да настану у практичној примени, као што су ситуација када апарат остане без неког од састојака за припрему напитка, као и када нема довољно новца да врати кусур кориснику. У првом случају је дата могућност повратка у стање ST\_CHECK\_CREDIT уколико нема материјала за прављење одабраног напитка, где може да се изабере неки други напитак из понуде, за које постоје састојци или да се добије новац назад. Такође апарат улази у стање ST\_ERROR уколико апарат нема довољно новца да врати кусур.

На основу VHDL кода направљен је симбол компоненте машине стања, који је коришћен у тест шеми реализованој у графичком едитору за потребе провере исправности рада симулацијом. Симулацијом је утврђено да написани код, као и сама тест шема функционишу сагласно задатој спецификацији, за три могућа случаја која могу да се десе у практичној примени.

Предложено решење би могло да нађе практичну примену, при чему би лако могло да буде прилагођено специфичним захтевима корисника.

# ИНДЕКС ПОЈМОВА

**A**

Апарат за кафу

Архитектура

**Б**

Библиотека

**В**

Вектор

**Д**

Д флип-флоп

Деклерација

Детектор успонске ивице

Дефинисање

Дијаграм

**Е**

Ентитет

**И**

Излази сигнали

Имплементација

Имплементација

Интегрисана кола

**К**

Код

**Л**

Логичке капије

**М**

Машина стања

Милијеви излази

Мурови излази

**О**

Објекти података

Оператори

Опис машине стања

**П**

Пиће

Порт

Променљиве

Процес

**С**

Самоуслужни апарат

Секвенцијалне мреже

Сигнал

Сигнал такта

Симбол

Симулација

Следеће стање

Стандард

Стање

**Т**

Тест шема

Тип података

Тренутно стање

**У**

Узлазна ивица

Улазни сигнали

Услови

**Ф**

Фаза

Функција

**Х**

Хардвер

**Ш**

Шематски приказ

**H**

HDL

**P**

PLD

**S**

ST\_CHECK\_CREDIT

ST\_ERROR

ST\_PROVIDE\_ PRODUCT

ST\_PROVIDE\_TOTAL\_REFUND

ST\_RESET

**V**

VHDL

# ЛИТЕРАТУРА

[1] <https://willicom.com/zanimljivosti.php>

[2] <https://www.elektronika.ftn.uns.ac.rs/uvod-u-mikroracunarsku-elektroniku/wp-content/uploads/sites/134/2018/03/Vezba-7-Opis-konacnih-automata.pdf>

[3] <https://www.kudaveceras.rs/vesti/3628/zasto-su-vending-aparati-odlicna-stvar>

[4] <https://www.ucg.ac.me/skladiste/blog_10555/objava_44831/fajlovi/VHDL_skripta%202019.pdf>

[5] <http://scindeks-clanci.ceon.rs/data/pdf/0354-6829/2006/0354-68290602157M.pdf>

[6] <https://www.scilab.org/signal-edge-detection>

[7] <https://www.youtube.com/watch?v=m2D4iYep_pQ>

[8]<https://www.infineon.com/dgdl/Infineon-Component_Edge_Detector_V1.0-Software%20Module%20Datasheets-v01_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0e9383de1ef4>

# изјава о академскoј честитости

**ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ**

|  |  |
| --- | --- |
| **Студент (име, име једног родитеља и презиме):** |  |
| **Број индекса:** |  |

Под пуном моралном, материјалном, дисциплинском и кривичном одговорношћу изјављујем да је завршни рад, под насловом:

1. резултат сопственог истраживачког рада;
2. да овaj рад, ни у целини, нити у деловима, нисам пријављиво/ла на другим високошколским установама;
3. да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица;
4. да сам рад и мишљења других аутора које сам користио/ла у овом раду назначио/ла или цитирао/ла у складу са Упутством;
5. да су сви радови и мишљења других аутора наведени у списку литературе/референци који је саставни део овог рада, пописани у складу са Упутством;
6. да сам свестан/свесна да је плагијат коришћење туђих радова у било ком облику (као цитата, прафраза, слика, табела, дијаграма, дизајна, планова, фотографија, филма, музике, формула, вебсајтова, компјутерских програма и сл.) без навођења аутора или представљање туђих ауторских дела као мојих, кажњиво по закону (Закон о ауторском и сродним правима), као и других закона и одговарајућих аката Високе школе електротехнике и рачунарства струковних студија у Београду;
7. да је електронска верзија овог рада идентична штампаном примерку овог рада и да пристајем на његово објављивање под условима прописаним актима Високе школе електротехнике и рачунарства струковних студија у Београду;
8. да сам свестан/свесна последица уколико се докаже да је овај рад плагијат.

У Београду, \_\_. \_\_. 202\_. године

Својеручни потпис студента

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_