УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Јожеф Шандор

Програмска подршка за генерисање испитних секвенци за микроконтролере

МАСТЕР РАД

Ментор:
Др Миодраг Ђукић

Нови Сад, 2019

Захвалан сам ментору проф. др Миодрагу Ђукићу за помоћ, мотивацију, стрпљење и правилно усмеравање приликом писања овог рада.

Посебно се захваљујем Милану Станкићу за несебичан приступ и изузетно корисне консултације и инспирацију, савете који су у многоме допринели квалитету програмског алата описаног у овом раду.

Такође, значајну захвалност изражавам и Ненаду Четићу за истрајну стручну помоћ и подршку.

На крају, велико хвала мојој породици, чија подршка ми је била најзначајнија.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Мастер рад	
Аутор, **АУ**:		Јожеф Шандор	
Ментор, **МН**:		Проф. др Миодраг Ђукић	
Наслов рада, **НР**:		Програмска подршка за генерисање испитних секвенци за микроконтролере	
Језик публикације, **ЈП**:		Српски	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2019.	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад, Факултет тахничких наука, Трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**:		6/34/36/1/7/14	
Научна област, **НО**:		Електотехничко и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница, **ПО**:		Микроконтролери, валидација, тестне секвенце, програмски алати	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета Техничких наука, Трг Доситеја Обрадовића 6, 21000 Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		Овај мастер рад описује једно решење програмског алата за генерисање испитних секвенци у сврху конфигурисања микроконтролера. Генерисање излазног тока података се врши на основу улазних података организованих у облику XML садржаја. Циљ је да се представи механизам, који ће крајњим корисницима омогућити једноставан начин организовања улазних података. Излазни садржај представља уређену структуру низа података у облику који одговара Интеловој MCS-51 (8051) општонаменској фамилији микорконтролера.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	Др Милан Лукић	
	Члан:	Др Момчило Крунић	Потпис
	Члан:		
	Члан, ментор:	Др Миодраг Ђукић	

UNIVERSITY OF NOVI SAD **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master	
Author, **AU**:		Jozef Sandor	
Mentor, **MN**:		Miodrag Djukic, PhD	
Title, **TI**:		Tool for generating test sequences for microcontrollers	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2019.	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		6/34/36/1/7/14	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S**/**KW**:		Validation, XML schema, microcontrollers, test sequences	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		This paper gives one solution of a tool and a proposal for a source file format for generating test sequences for programming microcontrollers. The proposal is to implement sequences source in XML format. This way we should be able to avoid compiler usage for the user, with the additional benefits of a simplified and more human-readable mechanism for specifying sequences.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Dr Milan Lukić	
	Member:	Dr Momčilo Krunić	Menthor's sign
	Member:		
	Member, Mentor:	Dr Miodrag Đukić	

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Број:
	ЗАДАТАК ЗА МАСТЕР РАД	Датум:

(Податке уноси предметни наставник - ментор)

СТУДИЈСКИ ПРОГРАМ:	Рачунарство и аутоматика
РУКОВОДИЛАЦ СТУДИЈСКОГ ПРОГРАМА:	Проф. др Миодраг Ђукић

Студент:	Јожеф Шандор	Број индекса:	E2 112/2017
Област:	Рачунарске технике и рачунарске комуникације		
Ментор:	Проф. др Миодраг Ђукић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;

НАСЛОВ МАСТЕР РАДА:

Програмска подршка за генерисање испитних секвенци за микроконтролере

ТЕКСТ ЗАДАТКА:

Имплементирати програмски алат за генерисање испитних секвенци у сврху конфигурисања микроконтролера. Анализирати захтеве, пројектовати механизам, и тестирати апликацију. Документовати процес развоја апликације.

Руководилац студијског програма:	Ментор рада:

Примерак за: □ - Студента; □ - Ментора

С А Д Р Ж А Ј

1. Увод

Најочигледније примере присуства и експоненцијалног развоја технологије у нашој непосредној близини можемо да пронађемо у областима као што су: мобилна телефонија, телевизија, различите информационе технологије, интернет и друго [1]. Врло често се човекова улога као актуатора у неком процесу запоставља и препушта се самом систему да управља [2], доноси одлуке и води рачуна о постојећим процесима. Тренутни правац развоја аутомобилске индустрије сведочи у прилог овој изјави [3].

Из сваког од наведених примера области можемо да издвојимо заједничке градивне компоненте као што су процесори, микроконтролери, меморијске јединице, различити екрани за приказ информација и др. Индивидуалан развој и константно усавршавање ових елемената представља основу и предуслов развоја целокупног система. Као крајњи резултат, омогућава нам се управљање далеко сложенијим процесима и у могућности смо да одговоримо на комплексније захтеве корисника.

Неизоставни елемент процеса управљања представља микроконтролер. Као склоп за рачунање, реализован је у облику интегрисаног кола који обједињује све неопходне компоненте како би могао самостално да функционише. Под неопходним компонентама подразумевамо интегрисане микропроцесоре, аналогне и дигиталне улазе/излазе, различите меморије, бројаче, тајмере и друге специфичне елементе неопходне за реализовање рада и ужу примену самог микроконтролера.

Функцијски гледано, микроконтролер ради у бесконачној петљи и на одређени скуп улазних података примењује задати алгоритам (програм) и генерише, подешава излаз.

Микроконтролери налазе све већу примену у различитим областима израде софтверски контролисаних уређаја и система. Будући да су намењени за управљачко интензиван апликациони домен, микроконтролери поседују широк скуп инструкција за манипулисање подацима на ниском нивоу, нивоу бита. За верификацију микроконтролера се користи код и подаци, који демонстрирају рад скупа инструкција и функционалност физичке структуре микроконтролера. Тестира се приступ меморији преко показивача, све подржане врсте прекида као и њихових приоритета, рада са регистрима, колима временске контроле и слично.

Предмет овог рада обухвата израду програмског алата за генерисање испитних секвенци (низа података) који се користи у сврху конфигурисања односно верификације микроконтролера. Генерисање излазног тока података се врши на основу улазних података организованих у облику *XML* садржаја. Циљ је да се овим представи механизам, који ће крајњим корисницима омогућити једноставан и интуитиван начин организовања улазних података. Излазни садржај представља уређену структуру низа података у облику који одговара Интеловој MCS-51 (8051) општенаменској фамилији микорконтролера. Апликација се покреће из командне линије и заснована је на *.NET* технологији.

Овај рад је сачињен од шест поглавља. У наредном поглављу дат је преглед стања у области и описане су кључне теоријске основе које су потребне за разумевање концепта и имплементације. Између осталог разматра се предложена софтверска и хардверска архитектура. Анализом проблема се бавимо у трећем поглављу, док је у четвртом поглављу дат предлог и опис решења. Објашњени су главни елементи који доприносе квалитету и поузданости изабраног процеса генерисања испитних секвенци. Пето поглавље даје опис испитивања и верификације решења. У шестом поглављу, у оквиру закључка дат преглед имплементираног програмског алата и идеје за даљи развој и проширење уз квалитативну анализу процеса.

2. Теоријске основе рада

У овом поглављу је дат детаљнији опис технологија које су коришћене за реализацију програмског алата. У основи, пројекат је креиран *Visual Studio* програмским пакетом, који између осталог, комбинује програмски језик *C#* са *.NET* технологијом. Овим је покривен обиман скуп постојећих класа организованих у облику именованих простора, који се користе у изради различитих типова апликација и олакшавају сам процес програмирања. Коришћењем овакве архитектуре имамо имплементирану и подршку за манипулацију *XML* датотекама у оквиру поменутог програмског пакета.

2.1. XML (eXtensible Markup Language)

Представља проширив језик за означавање података и документа. Дизајниран је за потребе описа, преноса и чувања података и настао је као дериват *SGML*-а *(Standard Generalized Markup Language)*. Препорука за *XML* је издата 1998. године [4] и ту је описана верзија 1.0, која је још увек актуелна.

XML омогућава да се текстуалном документу додају метаподаци, информације о тексту и његовој структури. Ове информације су укључене у сам текст на такав начин да се могу идентификовати и синтаксно разликовати од полазног садржаја [5]. Иако је првенствено намењен за програмску обраду, *XML* формат је читљив и људима. Пример *XML* садржаја се може видети на лисингу 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<podaci>
	<kontakt tip="osoba">
		<ime>Jozsef Sandor</ime>
		<telefon>0693000006</telefon>
		<email>sandorjozef@yahoo.com</email>
	</kontakt>
</podaci>
```

Листинг 1: Пример *XML* садржаја

Уколико се планира трајније представљање података у електронској форми, онда се тешко могу заобићи *XML*-технологије. Све *XML*-технологије су занимљиве и пружају велике могућности у опису података. Како је *XML* метајезик, не постоји предефинисан скуп дозвољених речи (енгл. *tag*) којима се описује структура података у документима. Сам аутор, у зависности од потребе и конкретне примене одређује ове

основне градивне јединице, пратећи при том синтаксна и формална правила, која дефинишу општу форму добро дефинисаног и формираног *XML* документа [6].

Приликом размене *XML* докумената, потребно је да обе стране поштују заједнички договор о ознакама. Та заједничка конвенција, која дефинише дозвољени формат XML документа описује се као *Document Type Definition (DTD)* или новије *XML Schema* [7].

Као главне предности *XML* формата, могу се издвојити следеће:

- Робустан је, флексибилан и омогућава описивање врло комплексних структура података [8].
- Омогућава пренос података који је практично независан од хардверске и софтверске платформе [9].
- Постоји велики број готових решења (алата и библиотека) за рад са *XML*-ом, што значајно олакшава манипулисање *XML* документима, као и развој софтвера који користи *XML* формат за опис података.
- Слободан је (енгл. *non proprietary)* формат односно отворени стандард, чија је спецификација јавна и доступна свима. За коришћење нису потребне никакве лиценце [10].
- Постоје механизми верификације и валидације формата [11].
- Дизајниран је да буде читљив како рачунарима, тако и људима, што ауторима и корисницима значајно олакшава рад.

2.1.1. Градивни елементи *XML*-а

XML је конципиран као језик за описивање докумената и података. Под документима и подацима подразумевамо текстуалне документе или скупове података.

Градивни елементи *XML* документа су [12]:

- **Ознаке** (енгл. *tag*) које користимо за описивање података. Тагови нису унапред дефинисани. Стандард једино описује минимални скуп правила која документ мора задовољавати, међутим имена ознака нису унапред одређена. Корисници морају сами дефинисати дозвољене ознаке за означавање. Елементи започињу почетном ознаком. Пример `<podaci>` из листинга број 1, а завршавају са одговарајућом завршном ознаком `</podaci>`. Елемент може бити празан без свог садржаја. У том случају пишемо га са косом цртом на крају имена ознаке као

нпр. `<podaci/>`. Ако није празан, *XML* елемент мора имати завршну ознаку. Документ може имати само један коренски елемент - елемент унутар којег су сви остали елементи.

- Осим елемената и ознака, у *XML* документима најчешће срећемо **атрибуте**. Атрибути се придружују елементима са идејом да пруже додатне информације о својствима елемената. Елементи могу имати више атрибута, али унутар једног елемента не сме бити више атрибута с истим именом. У *XML* терминологији за атрибуте се каже да имају своју вредност, а за елементе се каже да имају свој садржај. Пример:

```
<kontakt tip="osoba">
```

- Осим елемената, у *XML* документу се могу појавити и **декларације**, на пример:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Ако је *XML* декларација присутна, мора се појавити на самом почетку докумена. Стандард дозвољава да *XML* документ буде без декларације.

- **Процесорске наредбе** - Наредбе се уписују у документ када желимо да имамо додатне податке за програме који ће обрађивати *XML*. У једном документу можемо имати више процесних наредби намењених различитим циљним програмима. Пример:

```
<?xml-stylesheet type="text/xml"href="settings.xsl" ?>
```

- **Коментари** - Сврха коментара је могућност додавања напомена у *XML* документ, а који се неће третирати као подаци. Пример коментара:

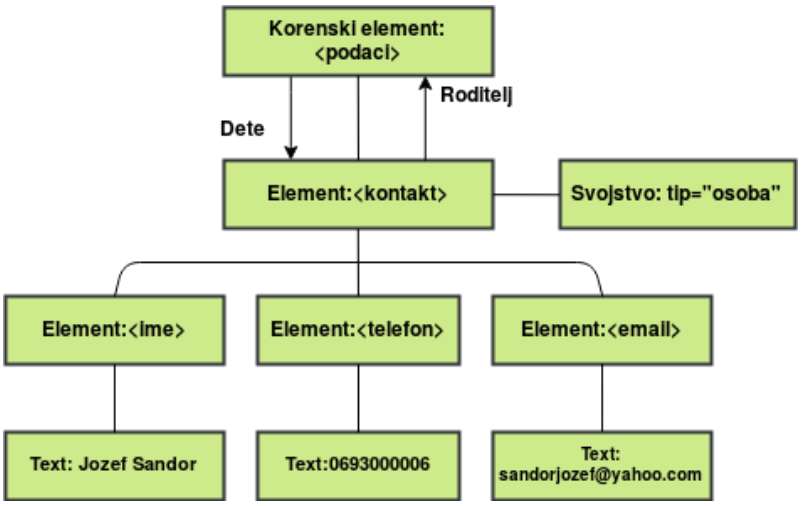
```
<!-- XML komentar-->
```

- ***DTD (Document Type Description)*** се односи на део *XML* документа који детаљније описује и ограничава дозвољени садржај документа. *DTD* је опционалан и не мора бити у истој датотеци са *XML* документом већ се може дефинисати у засебној датотеци омогућавајући на тај начин да више *XML* докумената референцира једну исту *DTD* датотеку. У новије време се уместо *DTD* датотеке користи *XML Schema*. Више о овоме у следећем поглављу.

- ***CDATA (Character Data)*** секције користимо када желимо да имамо делове текста који се неће ни на који начин обрађивати. У пракси се често користе за писање програмских исечака унутар *XML*-а.

- *XML* такође подржава **именске просторе** (енгл. *namespace*) који решавају проблем преклапања имена и омогућавају да у истом документу имамо иста имена са различитим значењем.

На слици број 1 је приказана блок структура *XML* датотеке дефинисане у горњем листингу.



Слика 1. Пример блок структуре *XML*-а

Према *XML* стандарду, на најнижем нивоу *XML* документ се састоји од низа *Unicode* знакова. *Unicode* стандард дефинше више начина кодирања знакова и данас представља најважнији знаковни стандард [13]. За записивање знакова се углавном користи више октета, за разлику од *ASCII* кодирања у којем се користимо само један октет. *Unicode* подржава све знакове најважнијих светских писама и граде различите делове *XML* документа. Најчешће су то елементи, под-елементи и атрибути као што је дато у горњим примерима.

2.1.2. Процесирање и валидација *XML* докумената

Пошто постоје правила за опис података у *XML*-у, постоје и средства помоћу којих се може проверити да ли су поштована правила *XML*-а за опис података. Ако су та правила задовољена у неком документу, онда се каже да је то *добро-формиран XML*-документ [14]. Да би могле да се примењују XML-технологије, документи с којима се оперише морају бити добро-формирани. Међутим, коришћењем додатних технологија

(DTD, XML-Schema), постоји могућност провере валидности описаних података. Овим се постиже, да кроз валидацију података, врше додатне провере података у погледу испуњености појединих захтева. Тиме се знатно смањује могућност да подаци с којима се оперише буду неисправни. Самим тим се и елиминишу грешке приликом коришћења тих података.

XML Schema има исту функцију као *DTD*, али је новија и опсежнија. Неке главне карактеристике [15]:

- Поставља услове и ограничења на садржај и структуру *XML* документа.
- Одређује правила која се морају поштовати да би се неки *XML* документ сматрао добро формираним.
- Дефинише речник ознака и граматику (дозвољени редослед елемената) који се могу појавити у документу.

Следећи листинг приказује шему која описује *XML* документ дефинисан у листингу број један. За тај документ кажемо да је добро формиран у односу на шему:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="podaci">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="kontakt">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="ime"/>
              <xs:element type="xs:string" name="telefon"/>
              <xs:element type="xs:string" name="email"/>
            </xs:sequence>
            <xs:attribute type="xs:string" name="tip"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Листинг 2: Пример *XML Shema* садржаја

Горња шема одређује да документ мора почети са елементом `<podaci>` која може имати произвољан број поделемената `<kontakt>`. Сваки елемент `<kontakt>` мора имати атрибут „`tip`“ и мора имати три поделемента `<ime>`, `<telefon>` и `<email>`.

2.1.3. Технологије везане за *XML*

XML као метајезик, који служи за опис других језика од самог почетка има огроман утицај на развој информатике и рачунарства. За врло кратак период развијени

су различити језици за опис структура и презентација на светској мрежи, за приказ мултимедијских као и математичких садржаја заснованих на *XML*-у [16]. У складу са тим, развијени су разни софтверски алати за примену ових језика, док су постојећа решења прилагођена коришћењу овог новог информатичког производа. И док се у почетку могло говорити само о *XML*-технологији, након разграњавања примене, може се говорити о читавом низу *XML*-технологија. Препознавајући потенцијал који крије ова нова технологија, велики произвођачи софтвера су главне своје производе засновали на њему, што је додатно допринело популарности.

XML је добро структуриран и у њему лако може да се оперише објектима. Подаци у *XML*-у се представљају у облику стабла, при чему сваки чвор у дрвету може да се третира као посебан објекат. Издвајањем чвора са свим гранама које полазе из њега, практично се издваја један објекат који се може на исти начин третирати као и објекти у објектно-оријентисаним језицима. На тај начин подаци описани помоћу *XML*-а могу се релативно једноставно обрађивати у објектно-оријентисаним језицима [17].

Кратак преглед и опис технологија које користе *XML* [18]:

- *DOM (Document Object Model)* омогућава обраду докумената описаних помоћу *XML*-а.
- *RDF (Resource Description Framework)* се односи на оперисање са метаподацима. Сматра се да *RDF* технологија треба да допринесе бољем разумевању података са светске мреже и олакша њихову обраду.
- *SVG (Scalable Vector Graphics)* је технологија која се односи на векторску графику и анимацију. Дефинише графику у *XML* формату.
- *SMIL (Synchronized Multimedia Integrated Language)* је језик везан за обраду мултимедијских докумената.
- *XHTML (Extensible Hypertext Markup Language)* је језик који представља *XML*-апликацију и служи за опис података на интернету.
- *SOAP (Simple Object Access Protocol)* платформски независтан комуникациони протокол базиран на *XML*-у. Користи се за размену информација између апликација преко *HTTP* протокола

- *WSDL (Web Sevices Description Language)* представља *XML* базиран језик за опис веб сервиса и приступ тим сервисима.
- *RSS (Really Simple Syndication)* је породица веб формата који се користе за објављивање садржаја интернет страница које се често мењају, као што су новински наслови; такође, може садржати и слике, аудио и видео садржаје и друге типове датотека.

2.2. Микроконтролери

Првенствено због своје цене, микроконтролери налазе све већу примену у свим областима у изради софтверски контролисаних уређаја и система. Друга велика предност микроконтролера, јесте чињеница да се могу програмирати у вишим програмским језицима типа Це [19]. Ово омогућава њихово програмирање па чак и без знања асемблерског језика.

Микроконтролер је мали рачунар, реализован у облику интегрисаног кола који обједињује све потребне компоненте, како би могао самостално да функционише. Ту спадају интегрисани микропроцесор, меморија, дигитални и аналогни улази/излази, тајмери, бројачи, осцилатори и други склопови (у зависности од врсте и намене микроконтролера) који су раније били сваки у свом интегрисаном чипу [20]. Микроконтролер се програмира да нормално ради у бесконачној петљи и за то време очитава улазе и подешава излазе у зависности од програма који му је задат.

Уопштене (заједничке) карактеристике микроконтролера су редом [21]:

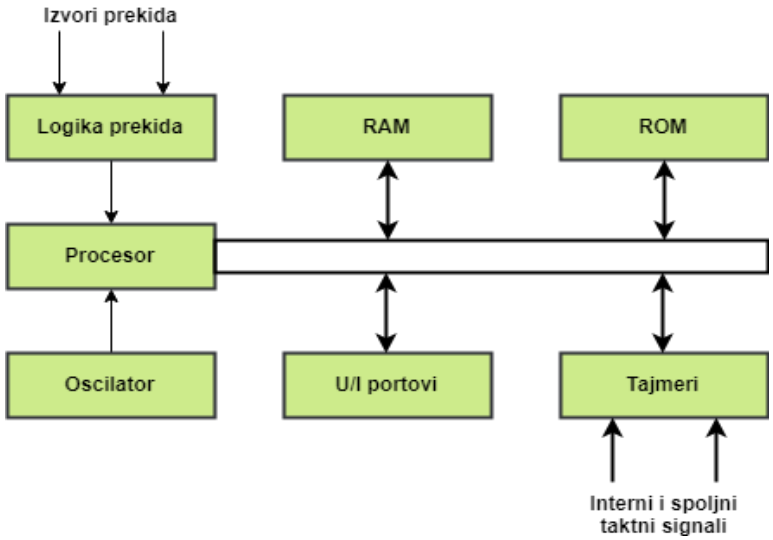
- Релативно мали радни такт реда 10MHz
- Мали број једноставних инструкција, реда величине око 100
- Радна меморија *(RAM)* реда *kB*
- Стална меморија са програмским кодом у *PROM* или *EPROM* изведби
- Бројачи различитих намена као сат, бројач импулса, *BCD* бројач и други
- Бројач за надзор исправног рада - *WDT (Watchdog Timer)*
- Улазно/Излазни канали за прихват и слање података
- *A/D* и *D/A* претварачи према намени, уобичајено 8-битни
- Широк распон напона напајања

2.2.1. Микроконтролер *INTEL 8051*

У фамилији *INTEL*-ових микроконтролера, *Intel 8051* је креиран наменски, да служи као процесорска јединица у системима дигиталног управљања. Унутар самог чипа се налазе *ROM* и *RAM* меморија, централна управљачка јединица, различити тајмери, бројачи као и паралелни и серијски улази односно излази [24]. Данас на

тржишту се може пронаћи велики број варијанти овог микроконтролера. Разликују се од основне верзије по броју и врсти периферијских компоненти, величини и типу уграђене меморије (нпр. репрограмибилан *ROM* у облику *FLASH* меморије), као и по механичкој.

Архитектура према меморијском моделу може бити Фон Нојманова, где су инструкције и подаци смештени у једну меморију или Харвард архитектура код које постоје физички одвојене меморије за инструкције и податке. Харвардска архитектура је новија и доноси низ предности као што су немогућност мешања инструкција и података, могућност коришћења различитих дужина инструкција и података као и скраћивање броја циклуса по операцији [25]. *Intel 8051* као и већина микроконтролера спада у микроконтролере Харвард типа. Основна варијанта чипа *Intel 8051* има структуру приказану на слици број 2.



Слика 2. Поједностављена структура чипа микроконтролера *Intel 8051*

Основне карактеристике микроконтролера *Intel 8051* су [26]:

- 8-битни *CPU*;
- 4kB интерне *ROM* или *EPROM* меморије за чување програма, са могућношћу проширења до 64kB спољашње меморије;
- 128 бајтова *RAM* меморије намењене за уписивање и читање података. У овој меморији се налазе 4 регистарске банке од по 8 регистара, као и стек меморија;
- 64kB адресног простора меморије података;

- 64kB адресног простора програмске меморије;
- 8-битни показивач стек меморије, који покрива интерну меморију;
- Два програмабилна 16-битна тајмера/бројача;
- Програмабилни серијски улаз/излаз;
- Четири 8-битна улазно/излазна прикључка;
- Тајмерски и улазно/излазни прекиди са два нивоа приоритета;
- 111 наредби;
- Аритметичко-логичка јединица (*ALU*) која може да извршава аритметичке операције сабирања, одузимања, множења и дељења, као и логичке операције;
- 256 адресибилних бита за рад са Буловом алгебром, од тога 128 бита у интерној меморији и 128 бита придружених постојећим интерним регистрима.

Овако синтетизован микроконтролер у многоме олакшава посао пројектанта хардвера, смањује укупно хардверско окружење и број спољашњих веза и на тај начин повећава поузданост система.

2.2.2. Меморијски модели микроконтролера

Intel 8051 поседује одвојену меморију за програм и податке, при чему сваки тип меморије поседује сопствени физички адресни простор и магистралу. Овим је омогућено симултано приступање програмској и меморији података, скративши тиме укупно време извршења програма.

Меморији за податке се брзо приступа јер се користи 8-битна адреса. За приступ интерној меморији се користе неколико различита меморијска типа [27]:

- *data* - означава да се приступа интерној меморији података уз директно адресирање, што омогућава брз приступ.
- *idata* - приступа се целој меморији за податке уз индиректно адресирање.
- *bdata* - означава да се приступа локацијама укупне величине 16 бајтова, које се могу адресирати по битовима.
- *xdata* – може се приступити било којој локацији унутар меморијског простора.

- *pdata* - означава де се приступа само једној страници величине 256 бајта од спољашње меморије за податке.

Меморијски модели *Intel 8051* микроконтролера се могу представити у неколико категорија, редом [28]:

- *Small model* - Код овог модела се све променљиве, подразумевано, налазе у унутрашњој меморији за податке *8051* система тј. користи се *data* меморијски тип. Код овог модела, променљивама се приступа на ефикасан начин. Међутим, сви објекти који нису експлицитно постављени у неком другом меморијском простору, морају да се уклопе у оквиру унутрашњег *RAM*-а који је сам по себи врло мали.
- *Compact model* - Код овог модела, подразумевамо да се све променљиве налазе у једној страници спољашње меморије за податке тј. користи се меморијски тип *pdata*. Овај модел може обезбедити максимално 256 бајтова променљивих, јер се користи индиректно адресирање кроз регистре *R0* и *R1*.
- *Large model* - Код овог модела се све променљиве налазе у спољашњој меморији за податке (до 64kB простора) тј. користи се *xdata* меморијски тип. Приступ меморији код овог модела ја у односу на остале моделе најспорији и неефикасан, нарочито код променљивих које садрже неколико бајтова.

Архитектура *Intel 8051* подржава неколико физички одвојених меморијских простора и делова за смештање програма. Постоје меморијски простори који омогућавају [29]:

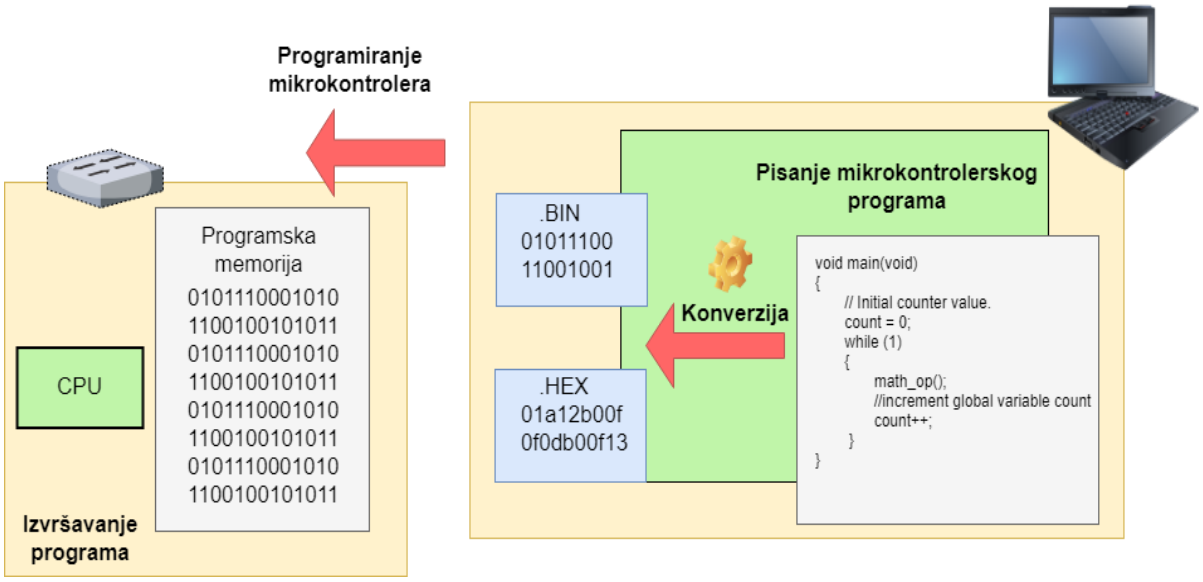
- читање али не и упис,
- упис и читање,
- упис и читање али брже у односу на остали део меморије.

2.2.3. Програмирање микроконтролера

Произвођачи микроконтролера углавном имају своја развојна окружења са којим подржавају програмске језике ниског нивоа као што су асемблер или Це. На тржишту постоје и развојна окружења других компанија која пружају кориснику могућност да са истим апликативним софтвером могу да развијају програме за мноштво различитих микроконтролера користећи један од понуђених програмерских

језика. Појављују се и виши програмски језици који се могу користити у ту сврху, као што су Це++, Јава или Пајтон.

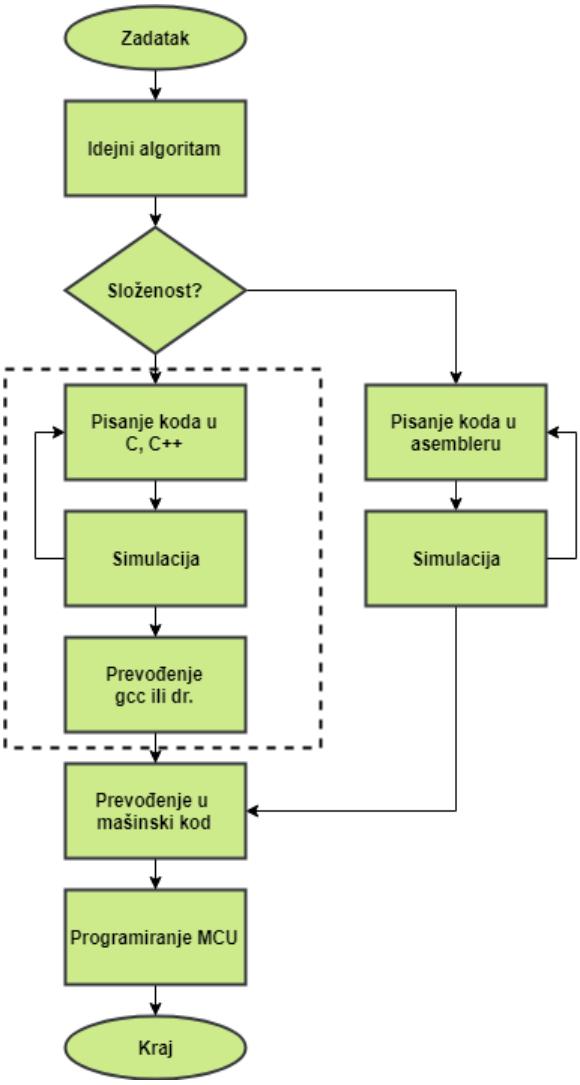
Упис генерисаног кода у микроконтролер се може извршити помоћу различитих решења. Најчешће се ради помоћу апликације и програматора чији је задатак да хексадецимални садржај преслика у меморију циљаног микроконтролера зарад извршавања [30]. Програматори могу бити засебни за професионалну употребу или интегрисани на прототипске штампане плоче. Овакве плоче садрже различита електронска кола и периферије, које се најчешће користе код уграђених система као што су *LED* диоде, тастери, дисплеји, интерфејси, сензори, итд. Новији микроконтролери су интегрисани са колом за исправљање којем приступа емулатор преко *JTAG*-а [31], што омогућава програмеру да исправи уграђени софтвер путем дебагера. Превођење са виших програмских језика на машински обавља компајлер који је најчешће део *IDE* окружења. Процес уписа је илустрован на слици број 3:



Слика. 3. Општи случај програмирања микроконтролера

3. Анализа проблема

Посматрајући уопштено процес програмирања једног микроконтролера, можемо да нацртамо следећи дијаграм:



Слика 4. Процес програмирања микроконтролера

У зависности од сложености решења бирамо на ком нивоу ћемо решити проблем, вишем или нижем, ближем машинском коду. Циљ рада и развијене апликације је да поједностави и апстрахује обележени део горњег дијаграма увођењем новог начина организовања изворних података у процесу генерисања испитних секвенци. Нагласак је на заобилажењу писања кода у вишем програмском језику и превођењу, док симулацију, откривање грешака и исправљање као саставни елемент тока рада у овом случају занемарујемо. Приликом описа нове архитектуре у наредним

поглављима, дат је упоредни приказ претходне имплементације испитних секвенци путем Це програмског језика и нове структуре у предложеном формату .

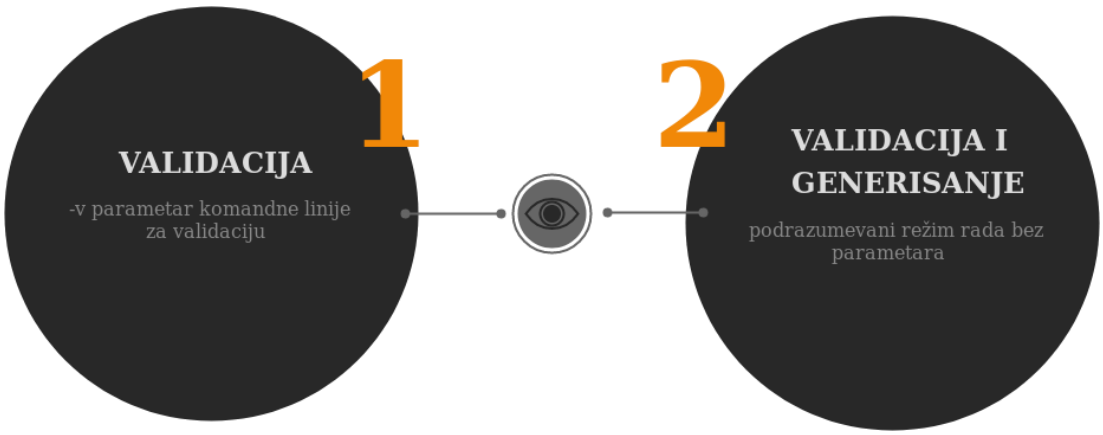
3.1. Организовање и приступ подацима

Задатак обухвата израду апликације која ће се покретати из командне линије и заснована је на *.NET* технологији. Израђен програмски алат за генерисање испитних секвенци користи се у сврху конфигурисања микроконтролера. Генерисање излазног тока података се врши на основу улазних података организованих у облику *XML* садржаја. У *XML*-у акценат је на опису података. Уопштено, *XML* и пратеће технологије карактерише дескриптивност уз прецизност описа и валидације података. Тиме се постиже олакшање процеса обраде података и смањење грешака приликом обраде.

Циљ је да се представи механизам, који ће крајњим корисницима омогућити једноставан и интуитиван начин организовања улазних података. Излазни садржај представља уређену структуру низа података у облику који одговара Интеловој *MCS-51 (8051)* општонаменској фамилији микорконтролера.

3.2. Случајеви коришћења

Дијаграм случајева коришћења представља све акције које корисник може да изврши и приказан је на слици број 5.



Слика 5. Дијаграм случајева коришћења

Креирани алат омогућава валидацију, као и генерисање испитних секвенци за појединачне, тако и за скуп улазних датотека који се налазе унутар једне фасцикле. У зависности од параметара које проследимо приликом покретања исте, дефинишемо

жељену операцију. У општем случају, коришћење апликације се своди на покретање команде следећег формата:

```
WriteSequencer [-v] <inFile> [<outFile>]
```

Листинг 3. Формат команде за покретање алата

Где су параметри редом:

1. `[-v]` Опциони праметар. Занемарује се ако је *outFile* параметар дефинисан. Уколико је присутан у пару са *inFile* параметром, указује на то да ће се искључиво валидација извршити над улазним подацима без генерисања излазних датотека. Потребно је назначити да се приликом процеса генерисања (*outFile* је дефинисан) валидација увек подразумева. Дакле можемо да радимо валидацију улазних датотека без генерисања излазних датотека, али не и генерисање излазних датотека без претходне валидације. Ова функционалност је нарочито корисна у току дефинисања испитних секвенци. Корисник је у могућности да утврди валидност креираних испитних секвенци већ од првих линија података.
2. `<inFile>` представља изворну *XML* датотеку која садржи низ дефиниција секвенци. Уколико наведемо путању фасцикле уместо путање датотеке, све *XML* датотеке унутар те фасцикле ће бити обрађене.
3. `[<outFile>]` опциони параметар који представља жељену излазну путању и назив саме излазне датотеке. Уколико је *inFile* директоријум, *outFile* такође мора да буде директоријум. У случају непостојања овог параметра у команди, излазна путања ће да буде једнака улазној пуној путањи (заједно са називом датотеке). Разлику ће представљати екстензија излазне датотеке у односу на улазну.

4. Имплементација

Коришћењем предложеног *XML* формата за организацију улазних података, отвара нам се могућност да поменуто писање кода у неком од виших програмских језика заменимо генерисањем испитних секвенци путем апликације, која ће извршити неопходне трансформације. На тај начин имамо могућност да избегнемо компајлирање, односно употребу било каквог другог потенцијално захтевног окружења.

Тестне секвенце у овом смислу подразумевају уређени низ различитих типова уписа. Скуп тренутно подржаних уписа [32]:

- ***Sparse Write***; јединични (индивидуалан) упис.
- ***Page Write***; садржи низ индивидуалних уписа удаљених за одређену вредност одступања (енгл. *offset*) од почетне адресе страничног уписа.
- ***Read Modify Write***; омогућава кориснику да модификује поље унутар одређеног регистра, док остатак регистра остаје непромењен. Имплементација овог уписа захтева коришћење маски одговарајуће ширине и вредности.
- ***Block Write***; упис блока података у жељени регистар.

Уз следеће додатне могућности, правила и ограничења:

- Сваки од уписа подржава рад са 8 или 32-битним подацима.
- Унутар сваког од типа уписа може да постоји и обезбеђен је механизам кашњења (енгл. *delay*).
- Унутар дефиниције секвенце може да конфигурише варијабилан број и тип уписа.
- Једна *XML* датотека може да садржи дефиницију више секвенци.
- Свака секвенца унутар датотеке се може једнозначно идентификовати називом секвенце.

4.1. Јединични упис (*Sparse Write*)

Представља најједноставнији облик индивидуалног регистарског уписа. У зависности од ширине података (8 или 32 бита) и постојања односно непостојања кашњења између уписа, разликујемо четири типа јединичног уписа.

Изражени у облику иницијално коришћених макроа:

```
SPARSE8_WRITE(REG_ADDR_1, 0x20, NO_EOS),
SPARSE8_WRITE_WITH_DELAY (REG_ADDR_1, 0x20, DELAY_128US, NO_EOS),
SPARSE32_WRITE(REG_ADDR_2, 0x97531357, NO_EOS),
SPARSE32_WRITE_WITH_DELAY (REG_ADDR_2, 0x97531357, DELAY_128US, NO_EOS)
```

Листинг 4. Јединични упис помоћу макроа

односно кроз нови формат:

```
<Write>
  <Address>REG_ADDR_1</Address>
  <Delay>NO_DELAY</Delay>
  <Value8 offset="0x0">0x20</Value8>
</Write>

<Write>
  <Address>REG_ADDR_1</Address>
  <Delay>DELAY_128US</Delay>
  <Value8 offset="0x0">0x20</Value8>
</Write>

<Write>
  <Address>REG_ADDR_2</Address>
  <Delay>NO_DELAY</Delay>
  <Value32>0x97531357</Value32>
</Write>

<Write>
  <Address>REG_ADDR_2</Address>
  <Delay>DELAY_128US</Delay>
  <Value32>0x97531357</Value32>
</Write>
```

Листинг 5. Јединични упис у *XML* формату

4.2. Странични упис (*Page Write*)

Може да садржи искључиво низ јединичних(енгл. *sparse*) уписа уз поштовање правила које намеће формат за овај тип уписа. Правила су редом:

- Адреса првог уписа је једнака пуној регистарској адреси задате у првом јединичном упису.

- Свака следећа адреса је померена у односу на почетну адресу и рачуна се маскирањем регистарске адресе са вредношћу `0xFF00`

Претходно коришћени макрои уз дефиницију подтипова уписа:

```
PAGE8_WRITE(REG_ADDR_1, 0x03, NO_EOS), 0x08,
              ADDR(0, REG_ADDR_1), 0x09,
PAGE8_WRITE_WITH_DELAY(REG_ADDR_2, 0x2, NO_EOS), 0x5,DELAY_1024US,
              ADDR(0, REG_ADDR_2), 0x09, DELAY_128US,
PAGE32_WRITE(REG_ADDR_3, 0x03, NO_EOS), DATA32(0x97531357),
              ADDR(0, REG_ADDR_3), DATA32(0x86420246)
PAGE32_WRITE_WITH_DELAY(REG_ADDR_4, 0X02, NO_EOS),
       DATA32(0x91731937), DELAY_512US,
              ADDR(0, REG_ADDR_4), DATA32(0x82642846), DELAY_0US,
```

Листинг 6. Странични упис помоћу макроа

Примери изражени кроз нови формат:

```
<PageWrite>
    <Write>
        <Address>REG_ADDR_1</Address>
        <Delay>NO_DELAY</Delay>
        <Value8 offset="0x0">0x08</Value8>
    </Write>
    <Write>
        <Address>REG_ADDR_2</Address>
        <Delay>DELAY_1024US</Delay>
        <Value8 offset="0x0">0x09</Value8>
    </Write>
    <Write>
        <Address>REG_ADDR_3</Address>
        <Delay>NO_DELAY</Delay>
        <Value32>0x97531357</Value32>
    </Write>
    <Write>
        <Address>REG_ADDR_4</Address>
        <Delay>DELAY_256US</Delay>
        <Value32>0x97531357</Value32>
    </Write>
</PageWrite>
```

Листинг 7. Странични упис у *XML* формату

4.3. Блоковски упис (*Block Write*)

За блоковски тип уписа у регистар је карактеристично да се упис врши у низ сукцесивних меморијских локација. Одмах по преносу прве речи блока преноси се следећа реч и тако редом до преноса свих речи блока. У нашој имплементацији разликујемо блоковске уписе, као и до сад, по ширини података (8 или 32-битни уписи), односно са и без прављења паузе(енгл. *delay*) између два захтева за упис. Такође, битно је напоменути да се у склопу блоковског уписа не могу комбиновати уписи елемената различитих ширина.

Примери уписа коришћењем макроа:

```
BURST8_WRITE(REG_ADDR_1, 0x5, NO_EOS), DATA8(0), DATA8(0x12),
              DATA8(0x34), DATA8(0x56),
BURST8_WRITE_WITH_DELAY(REG_ADDR_1, 0x2, NO_EOS), DATA8(0x08),
              DELAY_1024US, DATA8(0x09), DELAY_128US,
```

```
BURST32_WRITE(REG_ADDR_2, 3, NO_EOS), DATA32(0x98765432),
                DATA32(0x01357910), DATA32(0x024680),
BURST32_WRITE_WITH_DELAY(REG_ADDR_2, 0x2, NO_EOS), DATA32(0x98765432),
                DELAY_1024US, DATA32(0x01357910), DELAY_128US,
```

Листинг 8. Блоковски упис помоћу макроа

односно пример кроз предложени формат за податке ширине 8 бита:

```
<BlockWrite>
    <Address>REG_ADDR_1</Address>
    <Offset>0x0</Offset>
    <BlockElement8>
        <Delay>NO_DELAY</Delay>
        <Value8>0x01</Value8>
    </BlockElement8>
    <BlockElement8>
        <Delay>DELAY_128US</Delay>
        <Value8>0x23</Value8>
    </BlockElement8>
    <BlockElement8>
        <Delay>DELAY_128US</Delay>
        <Value8>0x45</Value8>
    </BlockElement8>
</BlockWrite>
```

Листинг 9. 8-битни блоковски упис у *XML* формату

као и податке ширине 32 бита:

```
<BlockWrite>
    <Address>REG_ADDR_1</Address>
    <BlockElement32>
        <Delay>DELAY_16384US</Delay>
        <Value32>0x02123456</Value32>
    </BlockElement32>
    <BlockElement32>
        <Delay>DELAY_64US</Delay>
        <Value32>0x789ABCDE</Value32>
    </BlockElement32>
    <BlockElement32>
        <Delay>DELAY_16384US</Delay>
        <Value32>0x0F123456</Value32>
    </BlockElement32>
</BlockWrite>
```

Листинг 10. 32-битни блоковски упис у *XML* формату

Може се приметити да у новом формату постоји могућност да корисник дефинише различито кашњење између појединачних уписа.

4.4. *RMW* (*Read-Modify-Write*) упис

За 8-битне, као и за 32-битне *RMW* уписе, циљ је селективно ажурирање података унутар регистара. Идеја иза 8-битног *RMW* уписа је да омогући кориснику да измени садржај поља који се протеже над подацима ширине мањег од једног бајта. Остатак података тог бајта припада другом пољу у склопу истог регистра. Дакле, уколико одређени податак представља садржај два различита поља, циљано поље треба да буде прочитано и ажурирано, док остатак податка, који припада другом пољу, треба да остане неизмењен. Слична правила важе и за 32-битни *RMW* упис. Кориснику је омогућено да измени поље које се протеже преко више бајтова унутар 32-битне речи. Селективно ажурирање података је постигнуто коришћењем маски при упису.

Постоје неколико ограничења која прате овај специфичан тип уписа:

- Маска мора да буде континуалан ток јединица. Валидне вредности маске су на пример: *0x1, 0x2, 0x3, 0x4, 0x6, 0x7, 0x8, 0xC, 0xE*. Очигледан пример невалидне маске је вредност *0x5*.
- Вредност *offset*-а у случају *RMW8* омогућава селекцију бајта унутар 32-битне речи.
- Вредности *value8* и *mask8* не смеју бити већи од *0xFF*

Испуњеност наведених услова и ограничења је проверена приликом валидације дефинисаних испитних секвенци.

Примери уписа помоћу макроа:

```
RMW8(REG_ADDR_1, 3, 6, 1, NO_EOS),
RMW8_WITH_DELAY(REG_ADDR_2, 2, 2, 2, DELAY_64US, NO_EOS),
RMW32(REG_ADDR_3, 3, 6, 0x56781234, NO_EOS),
RMW32_WITH_DELAY(REG_ADDR_4, 2, 26, 0x12345678, DELAY_64US, EOS),
```

Листинг 11. *RMW* упис помоћу макроа

Односно *XML*-а:

```
<ReadModWrite>
        <Address>REG_ADDR_1</Address>
        <Delay>NO_DELAY</Delay>
        <Value8 offset="0x0" >0x1</Value8>
        <Mask8>0xF0</Mask8>
</ReadModWrite>
<ReadModWrite>
        <Address>REG_ADDR_2</Address>
        <Delay>DELAY_64US</Delay>
```

```
        <Value8 offset="0x0" >0x2</Value8>
        <Mask8>0xF0</Mask8>
    </ReadModWrite>
    <ReadModWrite>
        <Address>REG_ADDR_3</Address>
        <Delay>NO_DELAY</Delay>
        <Value32>0x56781234</Value32>
        <Mask32>0x00FF00FF</Mask32>
    </ReadModWrite>
    <ReadModWrite>
        <Address>REG_ADDR_4</Address>
        <Delay>DELAY_64US</Delay>
        <Value32>0x12345678</Value32>
        <Mask32>0x00FF00FF</Mask32>
    </ReadModWrite>
```

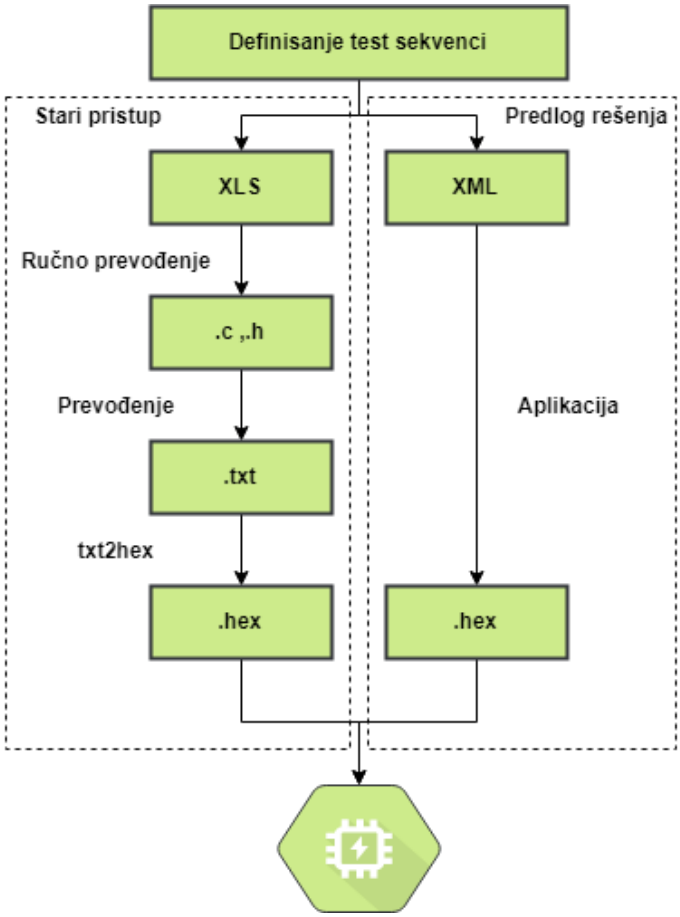
Листинг 11. *RMW* упис помоћу макроа

5. Анализа и верификација решења

Када говоримо о значају података за било које управљање, морамо приметити да су они врло често лоше организовани, што доводи до тешког управљања и лошег искоришћења. Раст количине података може додатно да компликује и отежа руковање истим. Неминовно долазимо до закључка да квалитетно организовани подаци представљају темељ и основу квалитетног решења.

5.1. Предности и мане

Предности предложеног решења генерисања испитних секвенци ћемо најлакше увидети тако што ћемо упоредити нови процес са до сад навикнутом процедуром генерисања. Графички приказано на слици број 6:



Слика 6. Упоредни дијаграм

Кораци старог приступа су редом:

- Креирање и унос неопходних елемената везаних за појединачне инструкције у *Excel* табеле. За сваки упис и секвенцу појединачно морамо дефинисати адресе регистара, припадајуће вредности, кашњења, одступања итд .
- Ручно превођење и поновно организовање секвенци у облик макроа дефинисаних у Це језику за сваки од типова уписа. Пример 8-битног страничног уписа дужине два елемената и 32-битног страничног уписа на крају секвенце:

```
const unsigned char code EXAMPLE (shutdown, SEQ_MODE_DISABLE)[] =
{
        PAGE8_WRITE(REG_ADDR_1 + 2, 0x03, NO_EOS), 0x08,
        ADDR(0, REG_ADDR_2 + 2), 0x09,
        SPARSE32_WRITE(REG_ADDR_3, 0x97531357, EOS)
};
```

Листинг 12. Пример секвенце

- Имплементација различитих трансформација над улазним подацима. За сваки тип уписа морамо дефинисати имплементацију макро функције. Пример:

```
#define PAGE8_WRITE(addr32, len, eos) \
SECTION_HEADER(VALID, WRITE, WITHOUT_DELAY, PAGE, X8, eos), \
LEN(len), \
ADDR(0, addr32), \
ADDR(1, addr32), \
ADDR(2, addr32), \
ADDR(3, addr32)
```

Листинг 13. Пример имплементације макроа

- Превођење и генерисање излазног тока у облику хексадецималног низа података.
- Програмирање чипа путем програматора.

Са друге стране имамо добро организовану *XML* датотеку и генерисану излазну *HEX* датотеку. Структура улазног података помоћу предложеног решења је организована у далеко интуитивнијем и прегледнијем формату у односу на садржај табеле и Це макроа, поготово кад се ради о комплекснијим секвенцама.

Неке предност *XML*-а:

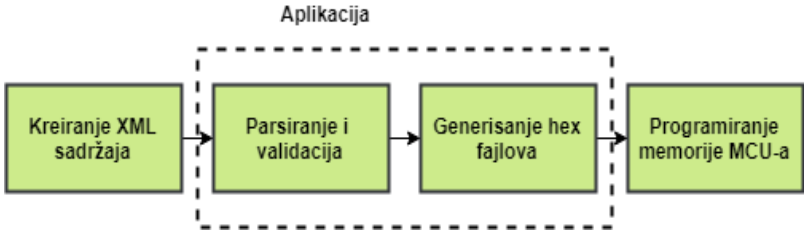
- Синтакса искључиво у текстуалном формату. У великој мери читљив и лак за писање и модификацију, чак и без посредства графичког алата.
- Синтакса није фиксна, дозвољава дефинисање нових типова података. Лако проширив уколико одлучимо да проширимо сет инструкција.
- Можемо да дефинишемо врсте маркирања које су дозвољене, неопходне или чак забрањене. Имамо могућност креирања XSD, DTD [33] шеме за валидацију формата креираног XML садржаја уз могућност пружања детаљних повратних информација и упутства кориснику о евентуалној начињеној грешци при креирању XML садржаја.
- Секвенцијална организација података и поновљивост креираних структура унутар секвенци је погодна за лаку имплементацију у XML формату, без великог губитка читљивости садржаја.

Такође, употребом новог формата имамо на располагању предности и могућност серијализације и десеријализације података. Језгро проблема превођења улазних података у одговарајући излазни ток смо овим свели на трансформацију и паковање улазних података у излазни ток по правилима дефинисаним за сваки од типова уписа. Претходно наведени пример секвенце у новој *XML* организацији:

```
<Sequences>
        <Sequence name="EXAMPLE">
        <PageWrite>
                <Write>
                        <Address>REG_ADDR_1</Address>
                        <Delay>NO_DELAY</Delay>
                        <Value8 offset="0x2">0x08</Value8>
                </Write>
                <Write>
                        <Address>REG_ADDR_2</Address>
                        <Delay>DELAY_1024US</Delay>
                        <Value8 offset="0x2">0x09</Value8>
                </Write>
        </PageWrite>
        <Write>
                <Address>REG_ADDR_3</Address>
                <Delay>NO_DELAY</Delay>
                <Value32>0x97531357</Value32>
        </Write>
        </Sequence >
</Sequences>
```

Листинг 14. *XML* формат тестне секвенце

На слици бр. 7 је приказан концептуалан систем програмирања микроконтролера помоћу апликације која генерише тестне секвенци помоћу предложене XML шеме.



Слика 7. Апстракција решења

Добијени резултат у облику низа хексадецималних података треба да остане непромењен, идентичан за оба наведена начина генерисања. Подржан је *Intel HEX* [34] као и *Motorola SREC* [35] формат чувања и паковања хексадецималног садржаја. Алат мимо *HEX* фајлова такође подржава и текстуални формат чувања генерисаних излазних података.

Имплементацијом генерисаних података у текстуалном формату остављамо простор за даљи развој и нови формат коришћења генерисаних података. Примера ради, генерисане секвенце се могу пресликати на почетне адресе одређених регистара путем текстуалних скрипти у некој од апликација вишег нивоа.

5.2. Тестирање и резултати

За проверу тачности реализованог решења и генерисаних података имали смо на располагању скуп предефинисаних очекиваних резултата у облику *.HEX* и текстуалних датотека. Референтни подаци за ову групу тестова су генерисани коришћењем постојећег алата. Упоређивањем излазих података утврдили смо да креирана апликација није правила никакве нежељене трансформације над улазним подацима. Преглед испитних случајева и број тестова дат је у табели 1.

Назив тестне групе	Број тестова
Јунит тестови	90
- Руковање улазним аргументима	15
- Валидација појединачних уписа	27
- Валидација секвенци	28
- Генерисање излазних датотека	20
Тестови за генерисање .HEX датотека	25

Табела 1. Преглед испитних случајева

6. ЗАКЉУЧАК

Задатак овог рада је била реализација програмске подршке за генерисање испитних секвенци за Интелову 8051 фамилију микрочипова. Одабрали смо XML формат за организовање улазних података због једноставности, прегледности, платформске независности развоја садржаја.

Реализована програмска подршка је развијена модуларно, тако да је остављена могућност за једноставно проширење и прилагођавање. Дефинисање додатних операција, додатних шаблона за опис операције, генерисање секвенци је изводљива уз минималне измене већ постојећег решења.

Интеграција са постојећим алатима за конфигурисање микроконтролера може бити остварена коришћењем постојеће комуникационе RPC спреге [36] или додавањем нових модула у постојећу апликацију који би омогућили комуникацију са физичком архитектуром.

Као даљи могући правац примене може да буде генерисање фајлова за мапирање адреса секвенци у оквиру меморије. Имплементација графичке корисничке спреге (графичког едитора) би омогућила једноставније креирање секвенци, коришћење претходно креираних секвенци или композитних инстукција и секвенци. Комбиновањем различитих елементарних уписа у композитне структуре можемо да креирамо комплексније наредбе и да додатно поједноставимо предложену организацију.

ЛИТЕРАТУРА

[1] Ian McNeil, *An Encyclopaedia of the history of technology,* Routledge, 29 West 35th Street, New York, 2002.

[2] Felix Kuhnert, Christoph Stürmer and Alex Koster, *Five trends transforming the Automotive Industry,* PricewaterhouseCoopers, 2017.

[3] A. Ferré, S. Alquézar, J. Fontanilles, J. Mestre, E. Montero, C. Daniel y M. Alves, "System on Chip and its Applications on Automotive Electronics", FISITA World Automotive Congress 2004, Barcelona, May 2004

[4] Bray, T. "Extensible Markup Language (XML), W3C Proposed Recommendation 10-February-1998", REC-xml-19980210, February 10, 1998.

[5] Bray, Tim, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and François Yergeau. "Extensible markup language (XML)." *World Wide Web Journal* 2, no. 4 (1997): 27-66.

[6] Pruitt, Steve, Doug Stuart, Wonhee Sull, and T. W. Cook. "The merit of XML as an architecture description language meta-language." *Microelectronics and Computer Technology Corporation* (1998).

[7] Lee, Jae-Shin, and Kyoung-Hoon Yi. *"Apparatus and method for parsing XML document by using external XML validator."* U.S. Patent 8,413,041, issued April 2, 2013.

[8] Beckett, Dave, and Brian McBride. "RDF/XML syntax specification (revised)." *W3C recommendation* 10.2.3 (2004).

[9] Bosak, Jon. "XML, Java, and the future of the Web." *World Wide Web Journal* 2.4 (1997): 219-227.

[10] Rezayat, Mohsen. "Knowledge-based product development using XML and KCs." *Computer-aided design* 32.5-6 (2000): 299-309.

[11] Larcheveque, J.M.H., Narendran, A., Sikchi, P., Levenkov, A., Ardeleanu, A., Shur, A., Catorcini, A., Selim, N.S. and Bath, K.S., Microsoft Corp, 2007. *Validation of XML data files*. U.S. Patent 7,296,017.

[12] Bray, Tim, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and François Yergeau. "Extensible markup language (XML) 1.0." (2008).

[13] Unicode Consortium. *The Unicode Standard, Version 2.0*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[14] Qin, J., Zhao, S. M., Yang, S. Q., & Dou, W. H. (2005, August). XPEV: A storage model for well-formed XML documents. In *International Conference on Fuzzy Systems and Knowledge Discovery* (pp. 360-369). Springer, Berlin, Heidelberg.

[15] Bex, Geert Jan, Frank Neven, and Jan Van den Bussche. "DTDs versus XML schema: a practical study." *Proceedings of the 7th international workshop on the web and databases: colocated with ACM SIGMOD/PODS 2004*. ACM, 2004.

[16] Laurent, Simon St, and Ethan Cerami. "*Building XML applications*." Vol. 11. New York: McGraw-Hill, 1999.

[17] Xiao, Renguo, et al. "Modeling and transformation of object-oriented conceptual models into XML schema." *International Conference on Database and Expert Systems Applications*. Springer, Berlin, Heidelberg, 2001.

[18] Qu, Yuzhong, et al. *"A Survey of XML and Related Technologies [J]."* Computer Engineering 12 (2000): 001.

[19] VanSickle, Ted. *Programming microcontrollers in C*. Newnes, 2001.

[20] Deshmukh, Ajay V. *Microcontrollers: theory and applications*. Tata McGraw-Hill Education, 2005.

[21] Vašek, Vladimír, et al. "Microcontrollers and modern control methods." *Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications: Modern Topics of Computer Science*. World Scientific and Engineering Academy and Society (WSEAS), 2008.

[22] Pearson, Ramesh S. Gaonkar-Prentice Hall. *Microprocessor Architecture Programming and Applications with the 8085, Atilim University Library, 2002: Microprocessor Architecture Programming and Applications with the 8085*. Vol. 1. Bukupedia, 2002.

[23] Sobotka, Zdenek. "Mikroprocesori i mikroracunala." *Tehnic ka knjiga, Zagreb* (1984).

[24] *MCS-51 Microcontroller Family User's Manual*, Intel Corporation, 1994.

[25] Deshmukh, Ajay V. *Microcontrollers: theory and applications*. Tata McGraw-Hill Education, 2005.

[26] Yeralan, Sencer, and Ashutosh Ahluwalia. *Programming and interfacing the 8051 Microcontroller*. Reading: Addison-Wesley, 1995.

[27] Jhang-Liang, L. I. N. "Integrated circuit for executing external program codes and method thereof." U.S. Patent No. 8,458,410. 4 Jun. 2013.

[28] Gupta, Gourab Sen. "*Embedded Microcontroller Interfacing: Designing Integrated Projects*." Vol. 65. Springer Science & Business Media, 2010.

[29] Parab, Jivan, Vinod G. Shelake, Rajanish K. Kamat, and Gourish M. Naik. "*Exploring C for microcontrollers: A hands on approach*." Springer Science & Business Media, 2007.

[30] Raj Kamal, "*Embedded Systems Architecture, Programming and Design",* Tata McGraw-Hill Publishing Company Limited, 7 West Patel Nagar, New Delhi, 2008.

[31] Jeppesen III, James Henry, and Kelly Sue St Clair-Hong. "JTAG interface system for communicating with compliant and non-compliant JTAG devices." U.S. Patent No. 5,708,773. 13 Jan. 1998.

[32] ARM® CoreLink™ AXI4 to AHB-Lite XHB-400 Bridge

[33] Bex, Geert Jan, Frank Neven, and Jan Van den Bussche. "DTDs versus XML schema: a practical study." *Proceedings of the 7th international workshop on the web and databases: colocated with ACM SIGMOD/PODS 2004*. ACM, 2004.

[34] *Hexadecimal Object File Format Specification* (PDF) (Revision A ed.). Intel. 1988-01-06.

[35] *Motorola M68000 Family Programmer's Reference Manual*, Motorola Inc., 1992

[36] Dejan Bokan, Nenad Četić, Jelena Kovačević, Marko Krnjetin, *Jedno rešenje komunikacione sprege u okviru okruženja za razvoj i kontrolu aplikacija za digitalne signal procesore*, ETRAN, Zlatibor, 2016