

## 7. Prototípus koncepció

54 – *Override*

Konzulens:

dr. László Zoltán

### Csapattagok:

Kriván Bálint	CBVOEN	balint@krivan.hu
Jákli Gábor	ONZ5G1	j_gab666@hotmail.com
Dévényi Attila	L1YRH0	devenyat@gmail.com
Apagyi Gábor	X8SG3T	apagyi.gabooo@gmail.com
Péter Tamás Pál	N5ZLEG	falconsaglevlist@gmail.com

2011. március 23.

# Tartalomjegyzék

<b>7</b>	<b>Prototípus koncepciója</b>	<b>4</b>
7.1.	Prototípus interface-definíciója . . . . .	4
7.1.1.	Az interfész általános leírása . . . . .	4
7.1.2.	Bemeneti nyelv . . . . .	4
7.1.3.	Kimeneti nyelv . . . . .	9
7.2.	Összes részletes use-case . . . . .	10
7.3.	Tesztelési terv . . . . .	11
7.4.	Tesztelést támogató segéd- és fordítóprogramok specifikálása . . . . .	11
7.5.	Napló . . . . .	11

## **Ábrák jegyzéke**

## 7. Prototípus koncepciója

### 7.1. Prototípus interface-definíciója

*[Definiálni kell a teszteket leíró nyelvet. Külön figyelmet kell fordítani arra, hogy ha a rendszer véletlen elemeket is tartalmaz, akkor a véletlenszerűség ki-bekapcsolható legyen, és a program determinisztikusan is tesztelhető legyen.]*

#### 7.1.1. Az interfész általános leírása

A prototípus szabványos ki- és bemeneten kommunikál a felhasználóval. Az elkészített prototípus program egy saját parancsrendszert használ. A parancs kiadása után a program végrehajtja azt és kiírja az eredményt a kimenetre. Az automatikus tesztelés elősegítése érdekében lehetőség van arra, hogy a parancsokat egy előre elkészített fájlból olvassa és a kimenetet fájlba mentse. A program az áramkört is fájlból olvassa. A tesztelés elősegítése érdekében elkészítettünk néhány áramkört, azonban a felhasználó a megadott áramkört leíró fájl specifikációja alapján saját áramkört is készíthet, majd tesztelhet.

#### 7.1.2. Bemeneti nyelv

##### 7.1.2.1. Felhasználói parancsok

A parancsokat a standard bemenetről, illetve fájlból olvassa be a program. Minden parancsot egy sorvége karakter zár le.

Megjegyzés: minden parancs ad visszajelzést a felhasználónak a végrehajtott eseményről, ennek formátuma a Kimeneti nyelv c. fejezetben olvasható.

*loadCircuit [file]*

- Leírás: A megadott áramkört betölti a szimulációs program.
- Megjegyzés: A file nevét kiterjesztés nélkül kell megadni.
- Opciók: -

*loadSettings [file]*

- Leírás: A jelenlegi áramkörhöz a megadott konfigurációs fájl betöltése.
- Megjegyzés: A file nevét kiterjesztés nélkül kell megadni.
- Opciók: -

*saveSettings [file]*

- Leírás: A pillanatnyilag használt konfiguráció fájlba mentése.
- Megjegyzés: A file nevét kiterjesztés nélkül kell megadni.
- Opciók: -

*switch [név]*

- Leírás: A megnevezett kapcsoló átállítása.
- Megjegyzés: -
- Opciók: -

*setSeqGen [név] [érték1, érték2, ...]*

- Leírás: A megnevezett szekvenciagenerátor az értékparaméterek szerint beállítódik.
- Megjegyzés: -
- Opciók: -

*getValue [név]*

- Leírás: A megadott áramköri elem kiírja az aktuális kimeneti értékét.
- Megjegyzés: -
- Opciók: a getValue -all parancs az összes áramköri elem kimenetének értékét kilistázza.

*step*

- Leírás: A parancs hatására lefut egy szimulációs ciklus, melynek két eredménye lehet:
  - véges lépésen belül stabilizálódik a rendszer, ekkor a kapcsoló(k), szekvenciagenerátor(ok) és ki-jelző(k) értéke(i) kiíródnak.
  - nem stabilizálódik az áramkör; hibaüzenet
- Megjegyzés: -
- Opciók: -

#### 7.1.2.2. Áramkör leíró fájlok nyelvtana

A konfigurációs fájlok \*.ovr kiterjesztésűek, ezekben adjuk meg a szimulálandó hálózat paramétereit. Egyszerű szövegfájl, melyben az értelmezendő parancsok soronként tagolódnak. A program feltételezi a konfigurációs fájl hibamentességét, sehol nem ellenőrizzük, hogy a bemenetnek van-e értelme! A fájl létrehozásához az alábbi parancsok, szintaxisok állnak rendelkezésre:

- X=...
  - Leírás: komponens létrehozás. Ezzel a paranccsal az egyenlőség jel után megadott komponenst hozzuk létre, melynek kimenetére ezek után az "X"-el hivatkozhatunk. Amennyiben több kimenetű komponensről beszélünk akkor az egyenlőség bal oldala egy tömböt jelent. Ebben az esetben az egyes kimenetekre a későbbiekben X[i]-vel hivatkozhatunk (a 0 és N-1 között).
  - Opciók: A lehetséges komponensek az implementált komponensek listájából választható, melyeknek paraméterül az egyes komponensekhez tartozó meghatározott paramétereket át kell adni.
    - \* OR(...)
    - \* AND(...)
    - \* INVERTER(...)
    - \* VCC(...)
    - \* GND(...)
    - \* MPX(...)
    - \* FFJK(...)
    - \* FFD(...)
    - \* LED(...)
    - \* 7SEG(...)
    - \* TOGGLE(...)

- \* SEQ(...)
- \* WIRE(...)
- \* NODE(...)
- OR(name,WIRE[n])
  - Leírás: vagy kapu létrehozása.
  - Opciók:
    - \* name: meg kell adni a kapcsoló nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* WIRE[n]: fel kell sorolni a kapu bemenetére kötött WIRE típusú változók neveit. N bemenetű kapu esetén ide N db WIRE kerül. A kapu számot nem kell megadni, azt a parser automatikusan észleli a megkapott paraméterek számából. Minimum 2 bemenetet meg kell adni.
  - Példa: OR(vagy1,w\_kapcs1\_vagy1,w\_kapcs2\_vagy1,w\_kapcs3\_vagy1) - három vezeték rákapcsolása a kapura, mely így egy három bemenetes vagy kapu lesz.
- AND(name,WIRE[n])
  - Leírás: "és" kapu létrehozása.
  - Opciók:
    - \* name: meg kell adni a kapu nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* WIRE[n]: u.a. mint a vagy kapu esetén.
  - Példa: AND(es1,w\_kapcs1\_es1,w\_kapcs2\_es1) - két vezeték rákapcsolása a kapura, mely így egy két bemenetes vagy kapu lesz.
- INV(name,WIRE\_in)
  - Leírás: inverter létrehozása.
  - Opciók:
    - \* name: meg kell adni a kapu nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* WIRE\_in: az inverter bemeneti vezetéket kell megadni. Csak egy bemenetű inverter létezik.
  - Példa: INV(inv1,w\_kapcs1\_inv1) - egy vezeték rákapcsolása a kapura.
- VCC(name)
  - Leírás: konstans igaz jel létrehozása.
  - Opciók:
    - \* name: meg kell adni a komponens nevét (egyedi azonosításra szolgál - a külvilág felé).
  - Példa: VCC(vcc1)
- GND(name)
  - Leírás: konstans hamis jel létrehozása.
  - Opciók:
    - \* name: meg kell adni a komponens nevét (egyedi azonosításra szolgál - a külvilág felé).
  - Példa: GND(gnd1)
- MPX(name,WIRE\_in[4],WIRE\_S[2])
  - Leírás: 4:1 multiplexer létrehozása.
  - Opciók:
    - \* name: meg kell adni a komponens nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* WIRE\_in[4]: meg kell adni négy WIRE-t, ami az egyes bemeneteket jelentik.

- \* WIRE\_S[2]: meg kell adni két WIRE-t, ami a két select jelet adja
  - Példa: MPX(mpx1,w\_in3,w\_in2,w\_in1,w\_in0,w\_s1,w\_s0)
- FFJK(name,WIRE\_clk,WIRE\_J,WIRE\_K)
  - Leírás: konstans hamis jel létrehozása.
  - Opciók:
    - \* name: meg kell adni a komponens nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* WIRE\_clk: meg kell adni egy WIRE-t, ami az órajel bemenet lesz
    - \* WIRE\_J: meg kell adni egy WIRE-t, ami az J bemenet lesz
    - \* WIRE\_K: meg kell adni egy WIRE-t, ami az K bemenet lesz
  - Példa: FFJK(ffjk1,w\_clk,w\_j,w\_k)
- FFD(name,WIRE\_clk,WIRE\_D)
  - Leírás: D flip-flop létrehozása.
  - Opciók:
    - \* name: meg kell adni a komponens nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* WIRE\_clk: meg kell adni egy WIRE-t, ami az órajel bemenet lesz
    - \* WIRE\_D: meg kell adni egy WIRE-t, ami az adat bemenet lesz
  - Példa: FFD(ffd1,w\_clk,w\_d)
- LED(name,WIRE\_in)
  - Leírás: LED létrehozása.
  - Opciók:
    - \* name: meg kell adni a komponens nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* WIRE\_in: meg kell adni egy WIRE-t, ami a LED bemenete lesz
  - Példa: LED(led1)
- 7SEG(name,WIRE\_D[8])
  - Leírás: 7 szegmenses kijelző létrehozása.
  - Opciók:
    - \* name: meg kell adni a komponens nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* WIRE\_D[8]: meg kell adni nyolc WIRE-t, ami a sorra a szegmenseket kapcsolja
  - Példa: 7SEG(7seg1,w\_d7,w\_d6,w\_d5,w\_d4,w\_d3,w\_d2,w\_d1,w\_d0)
- TOGGLE(name)
  - Leírás: kapcsoló létrehozása.
  - Opciók:
    - \* name: meg kell adni a kapcsoló nevét (egyedi azonosításra szolgál - a külvilág felé).
  - Példa: TOGGLE(kapcs1)
- SEQ(name,BITMINTA)
  - Leírás: szekvencia generátor létrehozása.
  - Opciók:
    - \* name: meg kell adni a komponens nevét (egyedi azonosításra szolgál - a külvilág felé).
    - \* BITMINTA: meg kell adni egy sorozatot, melyet a generátor egymás után kiad magából.

- Példa: SEQ(seq1,011000110)
- WIRE(name,WIRE\_in)
  - Leírás: vezeték létrehozása.
  - Opciók:
    - \* name: meg kell adni a vezeték nevét (egyedi azonosításra szolgál - a kívülág felé).
    - \* WIRE\_in: meg kell adni egy NODE-ot, vagy egy komponens kimenetet a vezeték bemenetének
  - Példa: WIRE(w1,X) - WIRE(w1,Node)
- NODE(name,WIRE\_in,out)
  - Leírás: csomópont létrehozása.
  - Opciók:
    - \* name: meg kell adni a csomópont nevét (egyedi azonosításra szolgál - a kívülág felé).
    - \* WIRE\_in: meg kell adni egy WIRE-t a csomópont bemenetének
    - \* out: meg kell adni, hogy a NODE-nak hány kimenete lesz
  - Példa: NODE(n1,w1,3) - három kimenetű elosztót hoz létre a w1 vezeték alapján
- Példa áramkör konfigurációs fájl Egy olyan minta hálózatot hozunk létre melyben található két kapcsoló egy és kapura kötve és az és kapu kimenete egy inverteren keresztül egy ledre kapcsolódik.
  - t1=TOGGLE(t1)
  - t2=TOGGLE(t2)
  - w\_t1\_es1=WIRE(w\_t1\_es1,T1)
  - w\_t2\_es1=WIRE(w\_t1\_es1,T2)
  - es1=AND(es1,w\_t1\_es1,w\_t2\_es1)
  - w\_es1\_inv1=WIRE(w\_es1\_inv1,es1)
  - inv1=INV(inv1,w\_es1\_inv1)
  - w\_inv1\_led1=WIRE(w\_inv1\_led1,inv1)
  - led1=LED(led1,w\_inv1\_led1)

### 7.1.2.3. Konfigurációs fájl nyelvtana

A konfigurációs fájlban minden sorban egybeállításnak kell szerepelnie, mely a következő egységekből áll:

- az elem neve
- egyenlőségjel
- az elem értéke (szekvencia generátor esetében több érték is lehet, ezeket vesszővel elválasztva kell megadni)

példa:

```
togle1=0
seqGen1=0,1,1,0,1
```



### 7.1.3. Kimeneti nyelv

A program történései, visszajelzése a standard kimeneten jelennek meg, illetve ezek fájlba is kiíródnak. A program minden parancs után visszajelzést ad a felhasználónak a végrehajtott eseményről. A fentebb definiált parancsokra a következő jelzéseket kapja a felhasználó:

#### *loadCircuit [file]*

Lehetséges kimenetek

- `load successful`
  - Leírás: a betöltés sikeres, amennyiben az áramkört tartalmazó fájl szintaktikája megfelel a Áramkör leíró fájlok nyelvtana c. fejezetnek.
- `load failed`
  - Leírás: a betöltés sikertelen, amennyiben az áramkört tartalmazó fájl szintaktikája nem felel meg a Áramkör leíró fájlok nyelvtana c. fejezetnek.

#### *loadSettings [file]*

Lehetséges kimenetek

- `load successful`
  - Leírás: az értékek betöltése sikeres, amennyiben a konfigurációs fájlban szereplő áramköri elemek megfeleltethetők az aktuális áramkörben szereplő elemekkel, illetve a megadott értékek helyesek.
- `load failed`
  - Leírás: az értékek betöltése sikeres, amennyiben a konfigurációs fájlban szereplő áramköri elem nem feleltethető meg az aktuális áramkörben szereplő elemek egyikével sem, illetve ha valamelyik érték helytelen.
  - Megjegyzés: azon elemek, melyek beállítására nem volt információ a konfigurációs fájlban automatikusan nullázódnak.

#### *saveSettings [file]*

Lehetséges kimenetek

- `save successful`
  - Leírás: a konfigurációs értékek sikeresen fájlba mentődtek.

#### *switch [név]*

Lehetséges kimenetek

- `[elem]: [érték]`
  - Leírás: az [elem] megadja a módosított kapcsoló nevét, míg az [érték] megmutatja, hogy milyen értékre változott az aktuális kapcsoló kimenete.

#### *setSeqGen [név] [érték1, érték2, ...]*

Lehetséges kimenetek

- `[elem]: [érték1, érték2, ...]`
  - Leírás: az [elem] megadja a módosított generátor nevét, míg az [érték1, érték2, ...] megmutatja, hogy milyen értékekre változott az aktuális generátor kimenete.

#### *getValue [név]*

Lehetséges kimenetek

- [elem]: [érték]
  - Leírás: az [elem] megadja a módosított kapcsoló nevét, míg az [érték] megmutatja, hogy milyen értékre változott az aktuális kapcsoló kimenete.
  - Megjegyzés: a `getValue` -all parancsra az összes elemet kilistázza a megadott formában új sor karakterrel elválasztva

*step*

## Lehetséges kimenetek

- `simulation successful`  
 [elem1]: [érték]  
 [elem2]: [érték]  
 ...
  - Leírás: a szimuláció sikeres, amennyiben véges lépésen belül stabilizálódni tud az áramkör. Ekkor a kapcsoló(k), szekvenciagenerátor(ok) és kijelző(k) értéke(i) kiíródnak.
- `simulation failed`
  - Leírás: a szimuláció sikertelen, amennyiben véges lépésen belül nem tud stabilizálódni az áramkör.

**7.2. Összes részletes use-case**

Use-case neve	Áramkör betöltése
Rövid leírás	Az áramkört leíró fájl betöltése
Aktorok	Felhasználó
Forgatókönyv	A <code>loadCircuit</code> parancsot használva betöltheti az áramkört leíró fájlt, amely a program követelményeinek megfelel

Use-case neve	Konfiguráció betöltése
Rövid leírás	Egy áramkör konfigurációjának betöltése
Aktorok	Felhasználó
Forgatókönyv	A <code>loadSettings</code> paranccsal betölt egy egyedi a konfigurációt az áramkörhöz, amely például tartalmazhatja a szekvencia generátorok által kiadott bitsorozatok vagy a kapcsolók állását.

Use-case neve	Konfiguráció mentése
Rövid leírás	Áramkör konfigurációjának mentése
Aktorok	Felhasználó
Forgatókönyv	A <code>saveSettings</code> parancs kiadásával menti az aktuális áramkör konfigurációját.

Use-case neve	Kapcsoló kapcsolása
Rövid leírás	Kapcsoló állásnak módosítás
Aktorok	Felhasználó
Forgatókönyv	Az adott áramkörben a neve alapján azonosított kapcsoló állásának módosítása a <code>switch</code> parancs használatával.

Use-case neve	Szekvenciagenerátor módosítás
Rövid leírás	Szekvenciagenerátor bitsorozatának megadása

Aktorok	Felhasználó
Forgatókönyv	Az adott áramkörben a neve alapján azonosított szekvenciagenerátor által kiadott bit-sorozat megadása a setSeqGen paranccsal.

<b>Use-case neve</b>	<b>Érték lekérdezése</b>
Rövid leírás	Egy, az áramkörben lévő alkatrész értékének lekérdezése
Aktorok	Felhasználó
Forgatókönyv	Az adott áramkörben a getValue parancs használatával a megadott nevű alkatrész értékének lekérdezése.

<b>Use-case neve</b>	<b>Áramkör szimulálása</b>
Rövid leírás	A betöltött áramkör szimulálása
Aktorok	Felhasználó
Forgatókönyv	A step parancs kiadásával szimulálja a betöltött áramkört.

<b>Use-case neve</b>	<b>Teszt eredményének ellenőrzése</b>
Rövid leírás	A program által generált kimenetet összehasonlítja a referencia kimenettel
Aktorok	Felhasználó
Forgatókönyv	A teszt lefutását követően egy script összehasonlítja a kapott eredményeket a várt eredményekkel.

### 7.3. Tesztelési terv

*[A tesztelési tervben definiálni kell, hogy a be- és kimeneti fájlok egybevetésével miként végezhető el a program tesztelése. Meg kell adni teszt forgatókönyveket. Az egyes tesztek elég informálisan, szabad szöveggént leírni. Teszt-esetenként egy-öt mondatban. Minden teszthez meg kell adni, hogy mi a célja, a proto mely funkcionálisát, osztályait stb. teszteli. Az alábbi táblázat minden teszt-esethez külön-külön elkészítendő.]*

<b>Teszt-eset neve</b>	...
Rövid leírás	...
Teszt célja	...

### 7.4. Tesztelést támogató segéd- és fordítóprogramok specifikálása

A program által generált kimeneti fájl és az elvárt eredményeket tartalmazó fájlok összehasonlítására a DiffUtils-ban (<http://www.gnu.org/software/diffutils/>) található cmp.exe-t fogjuk használni.

### 7.5. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2011.03.22. 12:00	2,5 óra	Apagyi G.	Prototípus áramkör leíró nyelvének definiálása
2011.03.22. 14:00	1,5 óra	Kriván B. Jákli G. Dévényi A.	Értekezlet: Specifikáció módosítása miatt szükséges szerű változtatások megbeszélése
2011.03.22. 20:00	1 óra	Jákli Gábor	Összes részletes use-case
2011.03.22. 22:00	1 óra	Dévényi A.	Felhasználói parancsok
2011.03.23. 14:00	1 óra	Dévényi A.	Konfigurációs fájl nyelvtana, kimeneti nyelv

Kezdet	Időtartam	Résztevők	Leírás
2011.03.23. 15:00	45 perc	Jákli G.	Új use case, 7.1.1 és 7.4