

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 28.

Tartalomjegyzék

7	Prototípus koncepciója	4
7.0.	Változtatások	4
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	8
7.2.	Összes részletes use-case	10
7.3.	Tesztelési terv	11
7.4.	Tesztelést támogató segéd- és fordítóprogramok specifikálása	12
7.5.	Napló	12

Ábrák jegyzéke

7. Prototípus koncepciója

7.0. Változtatások

A specifikáció módosulása miatt a következő változtatásokat kellett tenni a modellben. Bevezettük a Composite osztályt, mely egy kompozit áramköri elemet ír le. Mivel az áramkör is lényegében egy kompozit elem, ezért a Circuit osztály a Composite-ből öröklődik. Ha egy kompozit elemnek nincsen stacionárius állapota, akkor azt jelzi kifelé (az őt magába foglaló kompozitnak - amennyiben az áramkörrel van szó, akkor a szimulációt értesíti) és ezáltal az egész áramkörnek nem lesz stacionárius állapota. Az áramkör annyiból speciális kompozit, hogy nincsen be- és kimenete.

A másik új elem, mely bevezetésre került az oszcillátor. Ez egy olyan speciális LED, mely kijelzi az aktuális értékét, de lekérhető tőle az addig eltárolt értékek is. A memóriája véges, ha megtelik, az új érték a legrégebbi érték helyére íródik be.

7.1. Prototípus interface-definíciója

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 fájlból olvassa. A tesztelés elősegítése érdekében elkészítünk 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 7.1.3 alfejezetben olvasható.

loadCircuit <file>

- Leírás: A megadott áramkört betölti a szimulációs program. A szintaktikát lásd 7.1.2.2 fejezetet.

loadSettings <file>

- Leírás: A jelenlegi áramkörhöz a megadott konfigurációs fájl betöltése. A szintaktikát lásd 7.1.2.3 fejezetet.

saveSettings <file>

- Leírás: A pillanatnyilag használt konfiguráció fájlba mentése.

switch <név>

- Leírás: A megnevezett kapcsoló átállítása.

setSeqGen <név> <bit1>[<bit2>...]

- Leírás: A megnevezett szekvenciagenerátor az értékparaméterek szerint beállítódik.
- Megjegyzés: A szekvencia legalább 1 bitből áll.

check <név> | -all

- Leírás: A megadott áramkörü elem bemeneteinek és kimeneteinek kilistázása.
- Opció: a *check -all* parancs kilistázza az összes áramkörü elem bemenetét és kimenetét.

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

7.1.2.2. Az áramkör leíró fájl nyelvtana

Az áramkör leíró fájlban adjuk meg a szimulálandó hálózatot. Egyszerű szövegfájl, melyben az értelmezendő parancsok soronként tagolódnak. A program feltételezi, hogy ez egy, a Követelmények c. fejezetben megfogalmazott hálózatot ír le, azaz pl. nincsen lebegő ki- és/vagy bemenet.

A fájl egy sora az alábbi formátumú:

```
<név> = <komponens> ( <paraméter1> [, <paraméter2>, ...] )
```

Megjegyzés: a szóközők nem bírnak jelentéssel, ezek csak az olvashatóságot növelik.

Minden sor egy komponenszt ír le, az egyenlőség baloldalán található a komponens neve, mellyel a továbbiak során az adott komponensre, illetve annak valamelyik kimenetére hivatkozhatunk. A jobb oldalán pedig az adott komponens típusát határozzuk meg, illetve a paramétereket, melyek főképpen bemenetek:

- 0: konstans 0.
- 1: konstans 1.
- <név>[i]: egy adott komponens i. kimenete. Ha nem írjuk oda, hogy [i], akkor az alapértelmezett az, hogy a komponens 1. kimenetét kötjük oda. A név csak az angol ábc betűit és a számokat tartalmazhatja, illetve csak betűvel kezdődhet.
- Egyéb paraméter típus, ami a komponens létrehozásához kell (ilyen pl. a csomópontnak a kimeneti száma, illetve az oszcilloszkóp által eltárolható értékek száma), mely általában egy egész szám.

Ettől jelentősen eltér a kompozit definiálása (ennek felhasználása az áramkörben viszont ilyen alakú), ezt lásd később.

A lehetséges komponens típusok (és azok bemeneteinek) az alábbiak:

- `OR(in1, in2 [, in3, ...])`
 - Leírás: VAGY kapu létrehozása.
 - Paraméterek:
 - * N bemenetű kapu esetén ide N db bemenet kerül. A bemenetek számát nem kell megadni, azt a feldolgozó automatikusan észleli a megkapott paraméterek számából. Minimum 2 bemenetet meg kell adni.
 - Példa: `or = OR(kapcs1, kapcs2, kapcs3)` - három komponens rákapcsolása a kapura, mely így egy három bemenetes vagy kapu lesz.
- `AND(in1, in2 [, in3, ...])`
 - Leírás: ÉS kapu létrehozása.
 - Paraméterek:

- * N bemenetű kapu esetén ide N db bemenet kerül. A bemenetek számát nem kell megadni, azt a feldolgozó automatikusan észleli a megkapott paraméterek számából. Minimum 2 bemenetet meg kell adni.
- Példa: `and = AND(kapcs1, kapcs2)` - két komponens rákapcsolása a kapura, mely így egy két bemenetes vagy kapu lesz.
- `INV(in)`
 - Leírás: inverter létrehozása.
 - Paraméterek:
 - * `in`: az inverter bemenete.
 - Példa: `inv = INV(kapcs1)` - egy kapcsoló rákapcsolása az inverterre.
- `MPX(in1, in2, in3, in4, s1, s2)`
 - Leírás: 4:1 multiplexer létrehozása.
 - Paraméterek:
 - * `in1..in4`: a 4 adatbemenet.
 - * `s1, s2`: a 2 kiválasztó bemenet
 - Példa: `mpx = MPX(kapcs1, kapcs2, kapcs3, kapcs4, or1, and1)`
- `FFJK(clk, J, K)`
 - Leírás: J-K flip-flop létrehozása
 - Paraméterek:
 - * `clk`: órajel bemenet
 - * `J`: a J bemenet
 - * `K`: a K bemenet
 - Példa: `jk = FFJK(seqgen1, and1, and2)`
- `FFD(clk, D)`
 - Leírás: D flip-flop létrehozása.
 - Paraméterek:
 - * `clk`: órajel bemenet
 - * `D`: a D bemenet
 - Példa: `ffd = FFD(seqgen1, or1)`
- `LED(in)`
 - Leírás: LED létrehozása.
 - Paraméterek:
 - * `in`: led bemenete
 - Példa: `led = LED(kapcs1)`
- `7SEG(seg1, seg2, seg3, seg4, seg5, seg6, seg7)`
 - Leírás: 7 szegmenses kijelző létrehozása.
 - Paraméterek:
 - * `seg1..seg7`: szegmensek bemenetei
 - Példa: `display = 7SEG(seg1, seg2, seg3, seg4, seg5, seg6, seg7)`

- `TOGGLE()`
 - Leírás: kapcsoló létrehozása.
 - Példa: `t = TOGGLE()`
- `SEQGEN(seq)`
 - Leírás: szekvencia generátor létrehozása.
 - Paraméterek:
 - * `seq`: meg kell adni egy sorozatot, melyet a generátor egymás után kiad.
 - Példa: `seqgen = SEQGEN(011000110)`
- `NODE(in, n)`
 - Leírás: csomópont létrehozása.
 - Paraméterek:
 - * `in`: a csomópont bemenete
 - * `n`: csomópont kimeneteinek a száma
 - Példa: `node = NODE(kapcs1, 3)` - három kimenetű csomópontot hoz létre, melynek bemenete a `kapcs1`.
- `SCOPE(in, size)`
 - Leírás: oszcilloszkóp létrehozása.
 - Paraméterek:
 - * `in`: az oszcilloszkóp bemenete
 - * `size`: az oszcilloszkóp által tárolható értékek száma
 - Példa: `scope = SCOPE(kapcs1, 3)` - 3 értéket eltároló oszcilloszkóp, melynek bemenetére a `kapcs1` nevű komponens van kötve.

Kompozit definiálása:

A kompozitok definiálása lényegesen eltér az előzőektől, hiszen ott le kell írni a kompozit belsejét, majd erre a hálózatban hivatkozni lehet, ahogy a többi komponensre. Példa a kompozitra:

```
composite MY_COMPOSITE( x, y, z ) {

and = AND(x, y, z)
or = OR(and, y)

} (or)
```

A fenti példa egy olyan kompozitot ír le, melynek 3 bemenete van: `x`, `y` és `z`. Ezeket a bemeneteket egy 3-bemenetű ÉS és egy 2-bemenetű VAGY kapuban használjuk fel. A kompozitnak egy kimenete van, ezen pedig a VAGY kapu kimenete fog látszódni. Példa egy ilyen kompozit használatára:

```
toggle = TOGGLE()
comp = MY_COMPOSITE(toggle, 1, 1)
led = LED(comp)
```

Példa egy áramkör leíró fájlra:

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.

```

composite MY_COMPOSITE( x, y, z ) {

    and = AND(x, y, z)
    or = OR(and, y)

} (or)

t1 = TOGGLE()
t2 = TOGGLE()
comp = MY_COMPOSITE(t1, t2, 1)
led = LED(comp)

```

7.1.2.3. A konfigurációs fájl nyelvtana

A konfigurációs fájl minden sorában egy jelforrás beállítása szerepel, mely a következő egységekből áll:

- az elem neve
- egyenlőségjel
- az elem értéke (szekvencia generátor esetében a bitszekvenciát egybe kell megadni, lásd példa)

példa:

```

toggle1 = 0
seqGen1 = 01101

```

7.1.3. Kimeneti nyelv

A program történései, visszajelzése a standard kimeneten jelennek meg, illetve ezek fájlba is kiíródhatnak. 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 7.1.2.2 alfejezetben leírtaknak.
- load failed
 - Leírás: a betöltés sikertelen, amennyiben az áramkört tartalmazó fájl szintaktikája nem felel meg a 7.1.2.2 alfejezetben leírtaknak.

loadSettings <file>

Lehetséges kimenetek

- load successful
 - Leírás: az értékek betöltése sikeres, amennyiben a konfigurációs fájlban szereplő áramkörü elemek megfeleltethetők az aktuális áramkörben szereplő elemekkel, illetve a megadott értékek helyesek.
 - Megjegyzés: azon elemek, melyek beállítására nem volt információ a konfigurációs fájlban automatikusan nullázódnak.
- 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.

saveSettings <file>

Lehetséges kimenetek

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

switch <név>

Lehetséges kimenetek

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

setSeqGen <név> <bit1>[<bit2><bit3>...]

Lehetséges kimenetek

- `<név>: <bit1>[<bit2><bit3>...]`
 - Leírás: az [elem] megadja a módosított generátor nevét, míg az [érték1, érték2, ...] megmutatja, hogy milyen értékekre változott az aktuális generátor kimenete.

check <név> | -all

Lehetséges kimenetek

- `<név>:`
 - `in: <in1> [, <in2>, ...]`
 - `out: <out1> [, <out2>, ...]`
 - Leírás: kiírja a megadott elem ki és bemeneteit a fenti formátumban.
 - Megjegyzés: a `check -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 a megjelenítő(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 loadCircuit 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 loadSettings 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 saveSettings 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 switch 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 bitsorozat megadása a setSeqGen paranccsal.

Use-case neve	Elem vizsgálata
Rövid leírás	Egy, az áramkörben lévő alkatrész vizsgálata
Aktorok	Felhasználó
Forgatókönyv	Az adott áramkörben a check parancs használatával a megadott nevű alkatrész be- és kimeneteinek lekérdezése.

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

Use-case neve	Teszt eredményének ellenőrzése
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 tesztelés lehetőséget nyújt a program funkcióinak és menetének széleskörű vizsgálatára. Tesztelés során lehetőség nyílik a különböző tesztesetek kipróbálására. A teszt bemenetét bemeneti állományokból kapja, és a teszt eredményét kimeneti fájlban, és konzolon jeleníti meg. Ezáltal lehet összevetni a kiválasztott teszt várt, és tényleges eredményét.

Teszt-eset neve	Alap áramkör
Rövid leírás	Alap áramkör, mely kapcsolókból, és egyszerű nem visszacsatolt ÉS kapukból áll, kimeneti értékeket ledet jelzik
Teszt célja	Ez a teszteset leteszteli a kapcsoló, az ÉS kapu, és a led működését.

Teszt-eset neve	MPX-es áramkör
Rövid leírás	Olyan áramkör, mely kapcsolókból és MPX-ből áll, a kimeneti értékeket 7 szegmenses kijelző jelzi.
Teszt célja	Ez a teszteset leteszteli a MPX és a 7szegmenses kijelző működését.

Teszt-eset neve	Visszacsatolt stabil áramkör
Rövid leírás	Olyan áramkör, melyben egy visszacsatolás történik, de az áramkör stabil marad, tehát egy VAGY kapu visszakötve, kimenet értékét egy led jelzi.
Teszt célja	Ez a teszteset első sorban a visszacsatolás helyes működését teszteli le, de mivel visszacsatolás is történik és egyúttal a kimeneti értéket is megjelenítjük ledet, ezért csomópont is kell; így másodszorban a csomópont elem helyes működését is teszteli.

Teszt-eset neve	Visszacsatolt nem stabil áramkör
Rövid leírás	Olyan áramkör, melyben egy visszacsatolás történik, de az áramkör nem lesz stabil, mert egy visszacsatolt invertert tartalmaz.
Teszt célja	Ez a teszteset azt teszteli, hogy a szimuláció ilyen esetben leáll és ezt jelzi a felhasználónak.

Teszt-eset neve	Flip-flop-os áramkör
Rövid leírás	Olyan áramkör melyben szerepel egy flipflop, egy jelgenerátor, és egy oszcilloszkóp, melyre a flipflop kimenetét kötjük
Teszt célja	E teszteset során letesztelhetjük a jelgenerátor helyes működését és megvizsgálhatjuk, hogy a flip-flop helyesen lép-e felfutó élre, illetve helyesen működik-e, valamint az oszcilloszkóp helyesen tárolja-e az értékeket.

Teszt-eset neve	Kompozitos áramkör
-----------------	--------------------

Rövid leírás	Egy olyan áramkör, melyben szerepel egy kompozit elem, mely egy egyszerű áramköri hálózat tartalmaz.
Teszt célja	Ez a teszt eset a kompozit elem működését teszteli le, annak helyességét ellenőrizhetjük.

Teszt-eset neve	Kompoziton belüli kompozitos áramkör
Rövid leírás	Előző áramkörhöz hasonló áramkör, a különbség az, hogy a kompoziton belüli áramköri hálózat tartalmaz egy újabb kompozitot a többi elemen kívül.
Teszt célja	Ez az áramkör leteszteli, hogyan működik a program olyan esetben, mikor a kompozit további kompozitot tartalmaz, illetve a kompoziton belüli kompozit jól működik-e más elemekkel összekötve.

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	Résztevők	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
2011.03.23. 15:00	45 perc	Jákli G.	Új use case, 7.1.1 és 7.4
2011.03.26. 16:00	1,5 óra	Péter T.	Tesztelési terv és tesztesetek
2011.03.28. 11:30	2 óra	Kriván B.	Oszilloszkóp és kompozit elem felvétele a megfelelő fejezetekbe, illetve az osztálydiagram javítása.
2011.03.28. 12:00	30 perc	Dévényi A.	7.0-ás fejezet megírása.
2011.03.28. 12:30	45 perc	Jákli G.	A dokumentum átnézése, formázás javítása és helyesírás ellenőrzés.