# 4. Analízis modell kidolgozása 2

54 – *Override*

Konzulens:
dr. László Zoltán


Csapattagok:

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


2011. március 6.

# Tartalomjegyzék

# Ábrák jegyzéke

# 4. Analízis modell kidolgozása 2

## 4.1. Objektum katalógus

### 4.1.1. Simulation

A szimulációért felelős objektum. Létrehozza a szimulálni kívánt áramkört. Utasítja az áramkört több kiértékelési ciklus lefuttatásához, amíg az áramkörben van változás. Ha a változás megadott lépés szám limiten belül nem áll meg, tájékoztatja a felhasználót, hogy nincs stacionárius állapot.

### 4.1.2. State

A szimuláció állapotát megvalósító objektum. 3 állapota van: WORKING, READY, FAILED

### 4.1.3. Circuit

Az áramkör objektum. Ez az objektum hozza létre és köti össze egymással az áramköri elemeket. Törli az egyes elemek "már kiértékelt" flagjét a kiértékelési ciklus előtt, hogy ezáltal a ciklusban minden kimenet értéke frissülhessen. További feladata a kiértékelés elindítása az összes kijelzőre, mert a rendszer kiértékelése a kijelzők kiértékelésével kezdődik. A flip-flopokat utasítja arra, hogy mentsék el a jelenlegi kimenetüket, a saját belső memóriájukba. Megmondja, hogy az áramköri elemek értéke megváltoztak-e. Ezen kívül lépteti a jelgenerátorokat.

### 4.1.4. SequenceGenerator

Jelgenerátor, az áramkört felépítő egyik alapelem, kiértékelési kezdeményezés hatására az előre betáplált jelsorozat éppen aktuális elemét adja ki a kimenetén. Nincs áramköri bemenete. Mikor az áramkör kéri tőle, hogy lépjen, akkor a bitsorozat következő elemét fogja kiadni kimenetén.

### 4.1.5. Value

A logikai értékeket megvalósító objektum. Jelenleg két érték lehetséges: logikai igaz, logikai hamis.

### 4.1.6. AndGate

ÉS kapu, az áramkör egyik alapeleme. Bemeneteire kötött komponensek kiértékelését kezdeményezi, s a kapott értékek logikai ÉS kapcsolatát valósítja meg, melynek eredményét a kimenetén kiadja.

### 4.1.7. OrGate

VAGY kapu, az áramkör egyik alapeleme. Bemeneteire kötött komponensek kiértékelését kezdeményezi, s a kapott értékek logikai VAGY kapcsolatát valósítja meg, melynek eredményét a kimenetén kiadja.

### 4.1.8. Inverter

Invertáló, az áramkör alapelemei közé tartozik. A bemenetére érkező jel logikai negáltját valósítja meg, amit a kimenetén kiad.

### 4.1.9. Gnd

Föld, az áramkört felépítő egyik elem, állandó értéke logikai hamis. Bemenete nem létezik, így nem kezdeményez további kiértékeléseket.

#### 4.1.10. Vcc

Tápfeszültség, az áramkör egyik alapeleme, mely állandóan a logikai igazt adja ki a kimenetén.

#### 4.1.11. Led

Egy kijelző, az áramkör alapeleme, bemenetére kötött komponens kiértékelését kezdeményezi, és ezáltal az aktuális értékét egy a felhasználó számára érzékelhető módon kijelzi.

#### 4.1.12. Toggle

Kapcsoló, az áramkör egyik eleme, melynek aktuális értéke állítható, ez jelenik meg a kimenetén. Áramköri bemenete nincs.

#### 4.1.13. Value

A logikai értékeket megvalósító objektum. Jelenleg két érték lehetséges: logikai igaz, logikai hamis.

#### 4.1.14. FlipFlopD

D flip flopot megvalósító objektum. Csak akkor lép működésbe, mikor az órajelbemenetén a logikai érték hamisról igazra változik, ekkor az értékbemenettől függően változtatja a kimeneti értékét.

#### 4.1.15. FlipFlopJK

JK flip flopot megvalósító objektum. Csak akkor lép működésbe, mikor az órajelbemenetén a logikai érték hamisról igazra változik, ekkor az értékbemenetektől függően változtatja a kimeneti értékét.

#### 4.1.16. Mpx

4-1-es multiplexer áramköri építőelemet megvalósító objektum. Bemeneteire kötött komponensek kiértékelését kezdeményezi, a választó bemenet függvényében adja ki a kimeneten az egyik, vagy másik értékbemenetére kötött értéket.

#### 4.1.17. SevenSegmentDisplay

Hétszegmenses kijelző objektuma. Minden bemenete egy-egy szegmensért felelős, melyek 8-as alakban helyezkednek el.

### 4.2. Osztályok leírása

#### 4.2.1. AbstractComponent

Absztrakt osztály.

- Felelősség
  Egy komponens absztrakt megvalósítása, ebből származik az összes többi komponens. A közös logikát valósítja meg. A gyakran használt dolgokra ad alapértelmezett implementációt (összekötés, bemenetek kiértékelése stb.)
- Ősosztályok: (nincs)
- Interfészek: (nincs)
- Attribútumok

  - `private boolean changed`: Azt mutatja meg, hogy változott-e valamelyik kimenete a komponensnek. Ez a flag az `evaluate()` meghívásánál számolódik ki, vagyis azt jelzi, hogy két kiértékelés között változott-e a kimenet.
  - `protected boolean alreadyEvaluated`: "Kiértékelt" flag, ha ez be van billenve, akkor nem számolunk újra, csak visszaadjuk az előzőleg kiszámolt értéket.
  - `protected Value[] values`: Kimeneteken lévő értékek. Ez frissül az `evaluate()` meghívására, ha még nem volt kiértékelve.
  - `protected AbstractComponent[] inputs`: A bemenetekre kötött komponensek.

- Metódusok
  - `addTo(Circuit c)`: Meghívja az áramkör `add(AbstractComponent ac)` metódusát.
  - `Value[] evaluate()`: Komponens kimenetein lévő értékek újraszámolása (ha a kiértékelt flag, nincs beállítva) a bemenetek alapján. A kimeneteken lévő értékekkel tér vissza.
  - `void clearEvaluatedFlag()`: Töröljük a komponens "kiértékelt" flagjét.
  - `boolean isChanged()`: Changed flag lekérdező metódusa.
  - `void setInput(int inputPin, AbstractComponent component, int outputPin)`: Beállítunk egy bemenetet (adott bemeneti lábra rákötjük az adott komponens adott kimeneti lábát).

#### 4.2.2. AndGate

- Felelősség
  ÉS kapu, az áramkör egyik alapeleme. Bemeneteire kötött értékeken a logikai ÉS műveletet hajtva végre, és ennek eredményét adja ki a kimenetén.
- Ősosztályok: AbstractComponent
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

#### 4.2.3. Circuit

- Felelősség
  A komponensek létrehozása és összekötése a szimuláció elején, szimuláció által indított kiértékelési ciklus végrehajtása. Rajta keresztül léptethetőek a jelgenerátorok és véglegesíthető a FF-ok állapota.
- Ősosztályok: (nincs)
- Interfészek: (nincs)
- Attribútumok
  - `private Collection<AbstractComponent> components`: Összes áramköri komponens
  - `private Collection<DisplayComponent> displays`: Megjelenítő típusú komponensek (kijelző, 7-szegmenses kijelző)

  - `private Collection<SourceComponent> sources`: Jelforrás típusú komponensek (kapcsoló, jelgenerátor)
  - `private Collection<FlipFlop> flipFlops`: Flipflopok (D és JK flipflopok)
  - `private Collection<SequenceGenerator> sequenceGens`: Jelgenerátorok

- Metódusok
  - `void init()`: Áramkör inicializálása; komponensek létrehozása és összekötése.
  - `void add(AbstractComponent component)`: Komponens hozzáadása a `components`-hez. Ezt minden komponensre meg kell hívni.
  - `void add(SourceComponent sc)`: Jelforrás hozzáadása a `sources`-hoz.
  - `void add(FlipFlop ff)`: FlipFlop hozzáadása a `flipFlops`-hoz.
  - `void add(SequenceGenerator sg)`: Jelgenerátor hozzáadása a `sequenceGens`-hez.
  - `void add(DisplayComponent dc)`: Megjelenítő hozzáadása a `displays`-hez.
  - `void doEvaluationCycle()`: Egy kiértékelési ciklus lefuttatása. Az áramkörtől ezután lekérdezhető, hogy változott-e a rendszer állapota, azaz valamelyik komponens eltérő kimenetet ad-e, mint az előző ciklusban.
  - `boolean isChanged()`: Áramkör változásának lekérdezése. Igazzal tér vissza, ha van olyan komponens, ami azt jelzi magáról, hogy változott a kimenete az előző kiértékeléshez képest.
  - `void commitFlipFlops()`: A flipflopok jelenlegi kimenetének elmentése belső állapotnak, és az órajel bemenetén lévő érték eltárolása az éldetektálás érdekében.
  - `void stepGenerators()`: Jelgenerátorok léptetése.

#### 4.2.4. DisplayComponent

Absztrakt osztály.

- Felelősség
  Megjelenítő típusú komponensek ősosztálya.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs).
- Attribútumok
  - (nincs)
- Metódusok
  - `addTo(Circuit c)`: Meghívja az áramkör `add(DisplayComponent dc)` metódusát (és az ősosztály implementációját is – hiszen ugyanúgy kell regisztrálni a megjelenítőket is, mint a többi komponenst).

#### 4.2.5. FlipFlop

Absztrakt osztály.

- Felelősség
  Flipflopok ősosztálya, itt vannak leírva a flipflopok közös logikája.
- Ősosztályok: AbstractComponent

- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - `addTo(Circuit c)`: Meghívja az áramkör `add(FlipFlop ff)` metódusát.
  - `void commit()`: Az FF jelenlegi kimenetét és az órajel bemenetét elmenti. Előbbivel frissítjük a belső állapotot, utóbbi pedig az éldetektáláshoz kell. Ezt akkor kell meghívni, amikor az áramkör az adott áramköri bemenetekre stabil állapotba ért.
  - `boolean isActive()`: Számolhat-e az FF? Ezt kell ellenőrizniük a konkrét flipflop implementációknak, hiszen ekkor kellhet a belső állapottól eltérő állapotot kiadni.

#### 4.2.6. FlipFlopD

- Felelősség
  D flipflop, mely felfutó órajelnél beírja a belső állapotába az adatbemeneten lévő értéket. Kimenetén a belső állapota jelenik meg.
- Ősosztályok: AbstractComponent → FlipFlop.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

#### 4.2.7. FlipFlopJK

- Felelősség
  JK flipflop, mely felfutó órajelnél a Követelmények résznél leírt módon a J és K bemenetén lévő értéktől függően változtatja a belső állapotát. Kimenetén a belső állapota jelenik meg.
- Ősosztályok: AbstractComponent → FlipFlop.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

#### 4.2.8. Gnd

- Felelősség
  A "föld" komponens, mely állandóan a hamis értéket adja ki. Nincs bemenete.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs)

- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

#### 4.2.9. Inverter

- Felelősség
  Inverter alkatrész, mely invertálva adja ki a kimenetén a bemenetén érkező jelet.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

#### 4.2.10. Led

- Felelősség
  Egy LED-et reprezentál, mely világít, ha bemenetén igaz érték van.
- Ősosztályok: AbstractComponent $\rightarrow$ DisplayComponent.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - `Value getValue()`: Visszaadja a bemenetére kötött értéket.

#### 4.2.11. Mpx

- Felelősség
  4-1-es multiplexer, amely 4 adatbemenettel, 2 kiválasztó-bemenettel és 1 kimenettel rendelkezik. A kiválasztó-bemenetekre adott értéktől függ, hogy melyik adatbemenet értéke jelenik meg az adatkimeneten.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

#### 4.2.12. OrGate

- Felelősség
  VAGY kapu, az áramkör egyik alapeleme. Bemeneteire kötött értékeken a logikai VAGY műveletet hajtva végre, és ennek eredményét adja ki a kimenetén.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

#### 4.2.13. SequenceGenerator

- Felelősség
  Jelgenerátort reprezentál, amely a beállított bitsorozatot adja ki.
- Ősosztályok: AbstractComponent → SourceComponent.
- Interfészek: (nincs)
- Attribútumok
  - `private int index`: A bitsorozat egy indexe, ez határozza meg, hogy éppen melyik értéket adja ki a kimenetén.
  - `private Value[] sequence`: Tárolt bitsorozat
- Metódusok
  - `addTo(Circuit c)`: Meghívja az áramkör `add(SequenceGenerator sg)` metódusát (és az ősosztály implementációját is – hiszen ugyanúgy kell regisztrálni a jelforrásokat is, mint a többi komponenst).
  - `Value[] getValues()`: Jelgenerátor bitsorozatának lekérdezése
  - `void setValues(Value[] values)`: Jelgenerátor bitsorozatának beállítása
  - `void step()`: A jelgenerátor lép, a bitsorozat következő elemére ugrik. A következő léptetésig ez kerül kiadásra a kimeneteken.

#### 4.2.14. SevenSegmentDisplay

- Felelősség
  7-szegmenses kijelzőt reprezentál, melynek 7 bemenete vezérli a megfelelő szegmenseket, ezek világítanak, ha az adott bemenetre logikai igaz van kötve.
- Ősosztályok: AbstractComponent → DisplayComponent.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - `Value getSegment(int idx)`: Visszaadja az adott szegmenshez tartozó bemenetre kötött értéket.

#### 4.2.15. Simulation

- Felelősség
  Az áramkörön kiértékelési ciklusok futtatása az adott áramkör bemenetekre (kapcsolók állapota, jelgenerátorok jelenlegi értéke) nézve addig, amíg az áramkör nem stabilizálódik.
- Ősosztályok: (nincs)
- Interfészek: (nincs)
- Attribútumok
  - `private Circuit circuit`: Szimulált áramkör
- Metódusok
  - `boolean start()`: Szimuláció elindítása a jelenlegi áramköri bemenetekre (kapcsolók állapota, jelgenerátorok jelenlegi értéke). Amennyiben stacionárius állapot jött létre, léptetjük a jelgenerátorokat és elmentjük a flipflopok állapotát. Így újbóli hívásra már a következő időpillanatban érvényes áramköri bemenetekre lehet szimulálni az áramkört. A visszatérési érték a sikerességet jelzi (sikertelen, ha nincs stacionárius állapot).
  - `State getState(State state)`: Szimuláció állapotának lekérdezése.

#### 4.2.16. Simulation.State

Enumeráció.

- Felelősség
  Szimuláció állapotait írja le
- Ősosztályok: (nincs)
- Interfészek: (nincs)
- Attribútumok
  - `public static final State READY` A szimuláció kész a futásra. Ilyenkor hívható rajta a start() metódus.
  - `public static final State WORKING` Szimuláció éppen dolgozik, egy konkrét jelforrás-kombinációt alkalmazva szimulálja az áramkört.
  - `public static final State FAILED` A szimuláció leállt, mert az áramkörnek nincs stacionárius állapota. A start() metódus újra hívható (ha a bemenetek nem változnak, továbbra is le fog állni).
- Metódusok
  - (nincs)

#### 4.2.17. SourceComponent

Absztrakt osztály.

- Felelősség
  Jelforrás típusú komponensek ősosztálya.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs).

- Attribútumok
  - (nincs)
- Metódusok
  - `addTo(Circuit c)`: Meghívja az áramkör `add(SourceComponent dc)` metódusát (és az ősosztály implementációját is – hiszen ugyanúgy kell regisztrálni a jelforrásokat is, mint a többi komponenst).
  - `abstract Value[] getValues()`: Lekérhetjük a jelforrás értékeit. Ennek megvalósítása a konkrét implementációk feladata.
  - `abstract setValues(Value[] values)`: Beállítjuk a jelforrás értékét. Ennek megvalósítása a konkrét implementációk feladata. (pl. kapcsoló csak 1 elemű tömböt kaphat)

#### 4.2.18. Toggle

- Felelősség
  Kapcsoló jelforrás, melynek két állapota lehet; egyikben logikai igazat, másikban logikai hamist ad ki.
- Ősosztályok: AbstractComponent → SourceComponent.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - `Value[] getValues()`: Lekérjük a kapcsoló értékét (1 elemű tömb)
  - `void setValues(Value[] values)`: Kapcsoló állapotának változtatása, csak 1 elemű tömböt kaphat paraméterül.

#### 4.2.19. Value

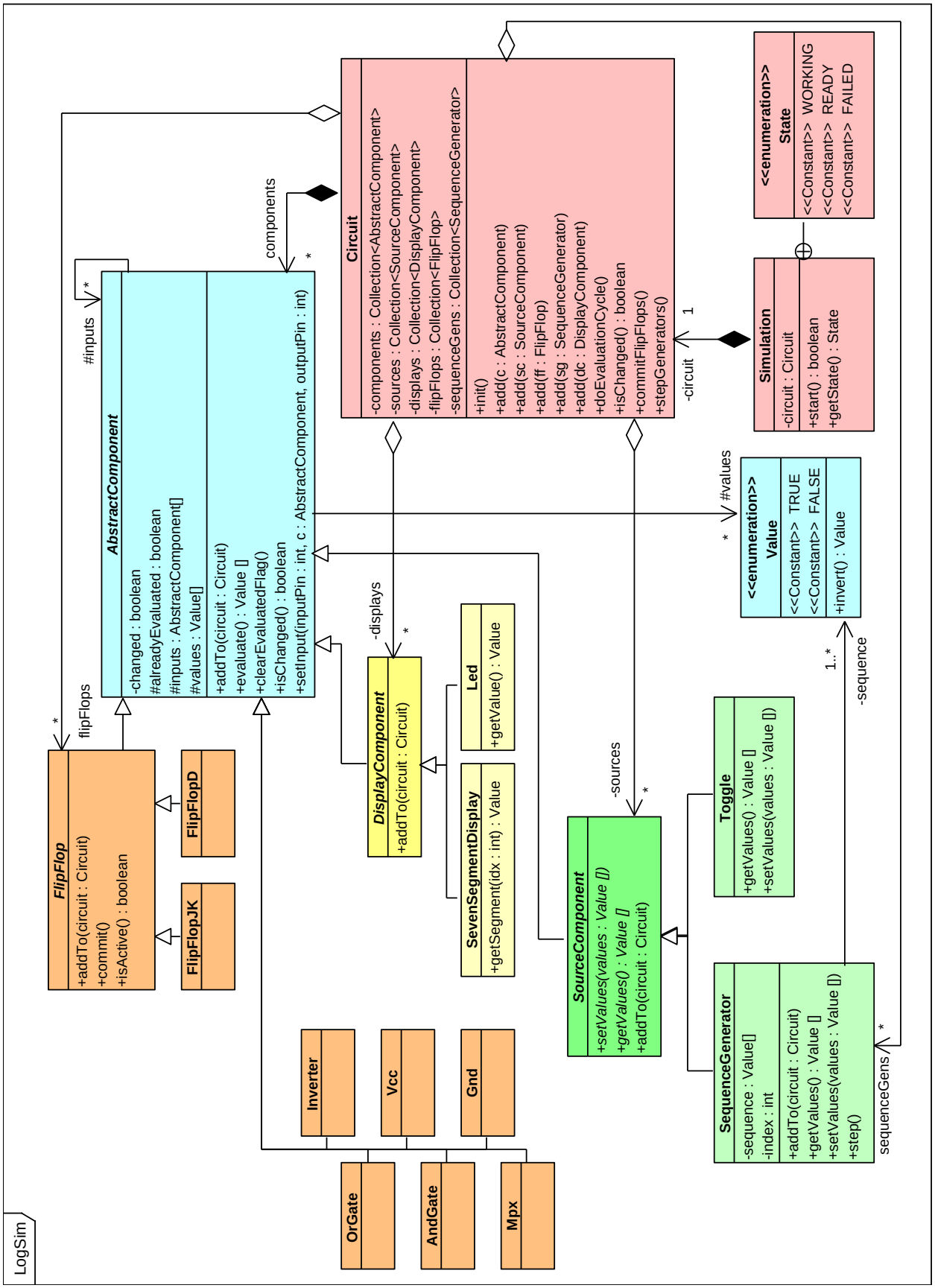
Enumeráció.

- Felelősség
  Az áramkörben előfordulható értéket reprezentál.
- Ősosztályok: (nincs)
- Interfészek: (nincs)
- Attribútumok
  - `public static final Value FALSE`
  - `public static final Value TRUE`
- Metódusok
  - `Value invert()`: Invertálja az adott értéket. Ennek addig van értelme, amíg 2 féle állapot fordulhat elő a rendszerben.

#### 4.2.20. Vcc

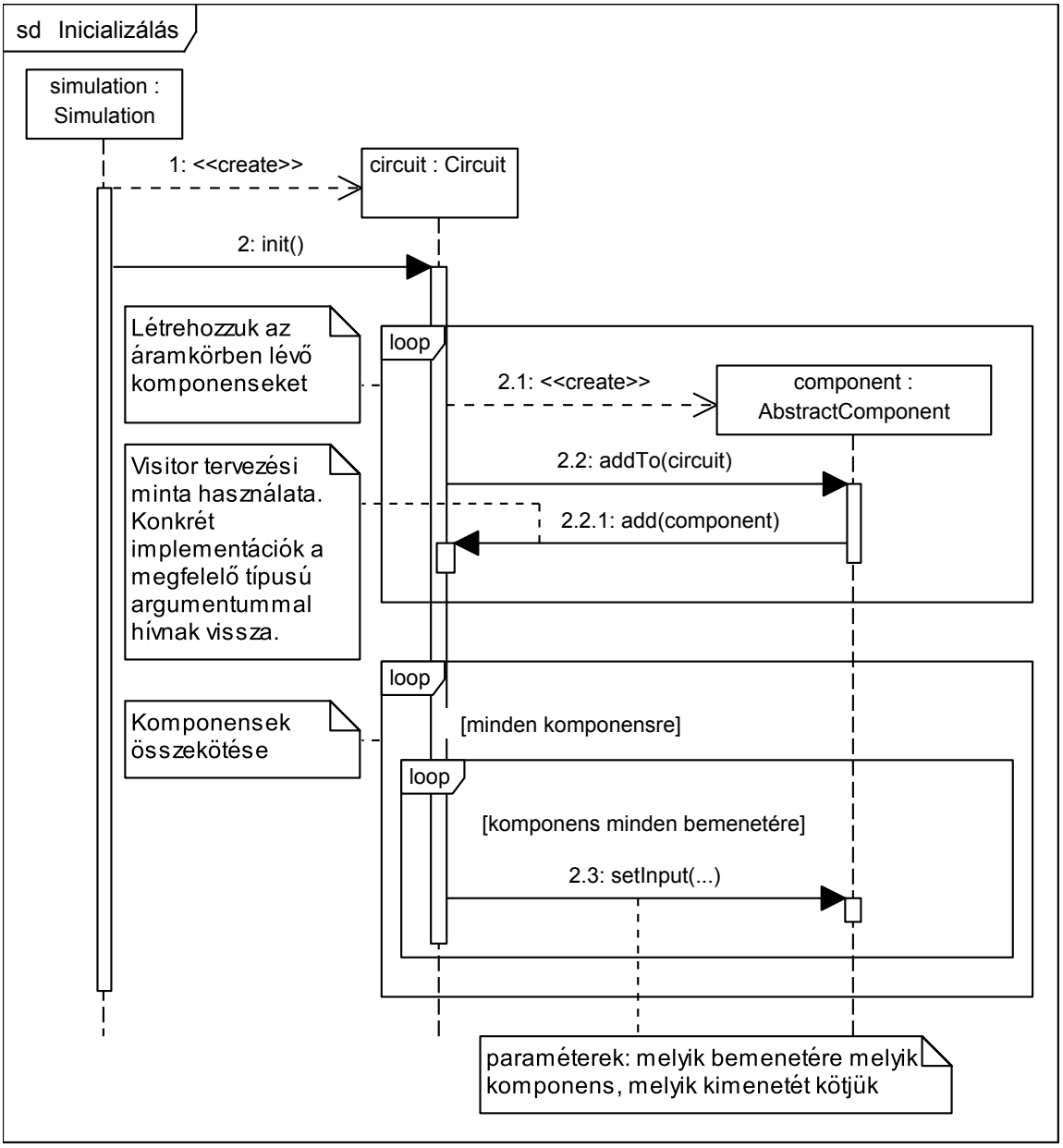
- Felelősség
  A tápfeszültés komponens, ami konstans igaz értéket ad. Nincs bemenete.
- Ősosztályok: AbstractComponent.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

### 4.3. Statikus struktúra diagramok

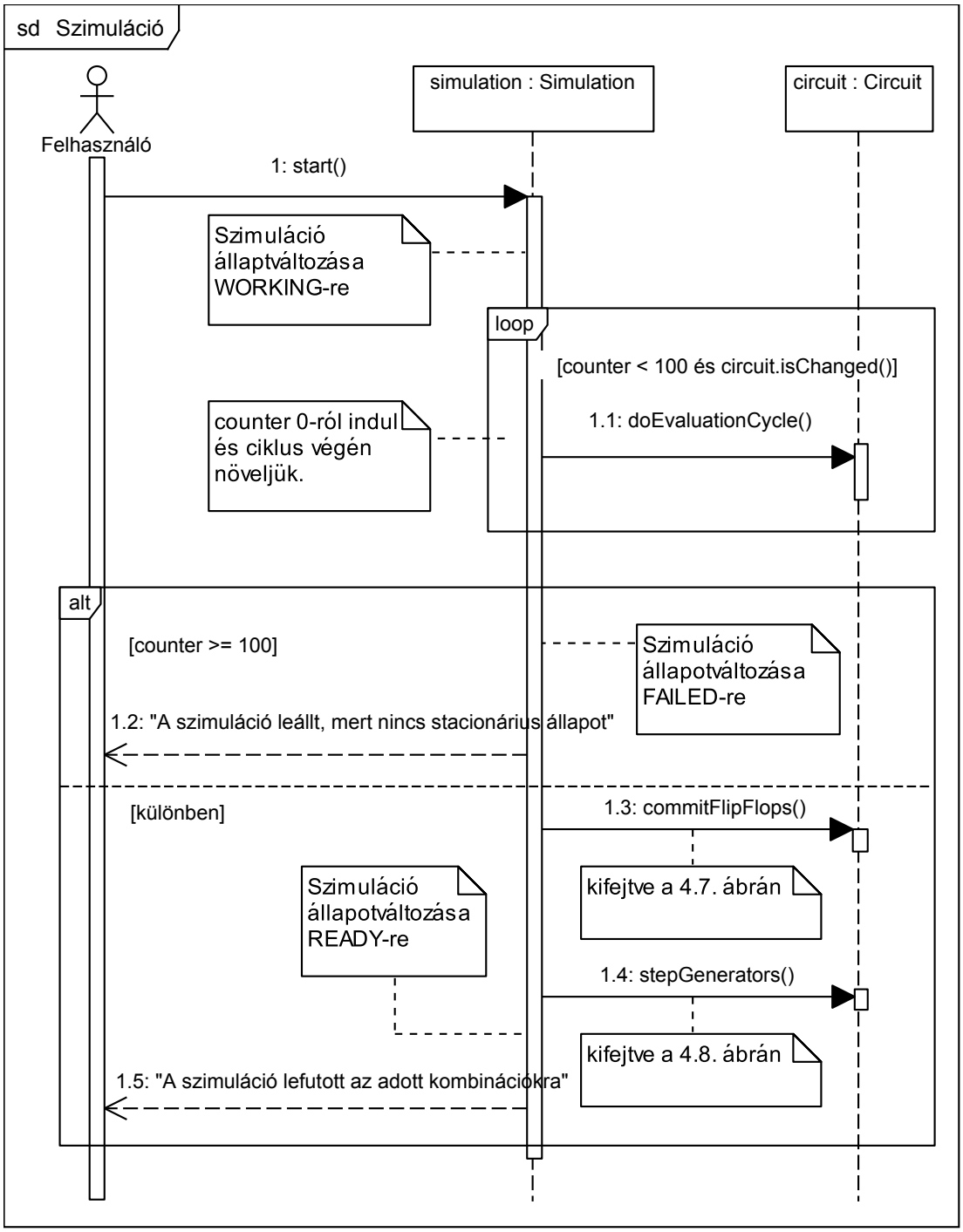



4.1. ábra. Statikus struktúra nézet

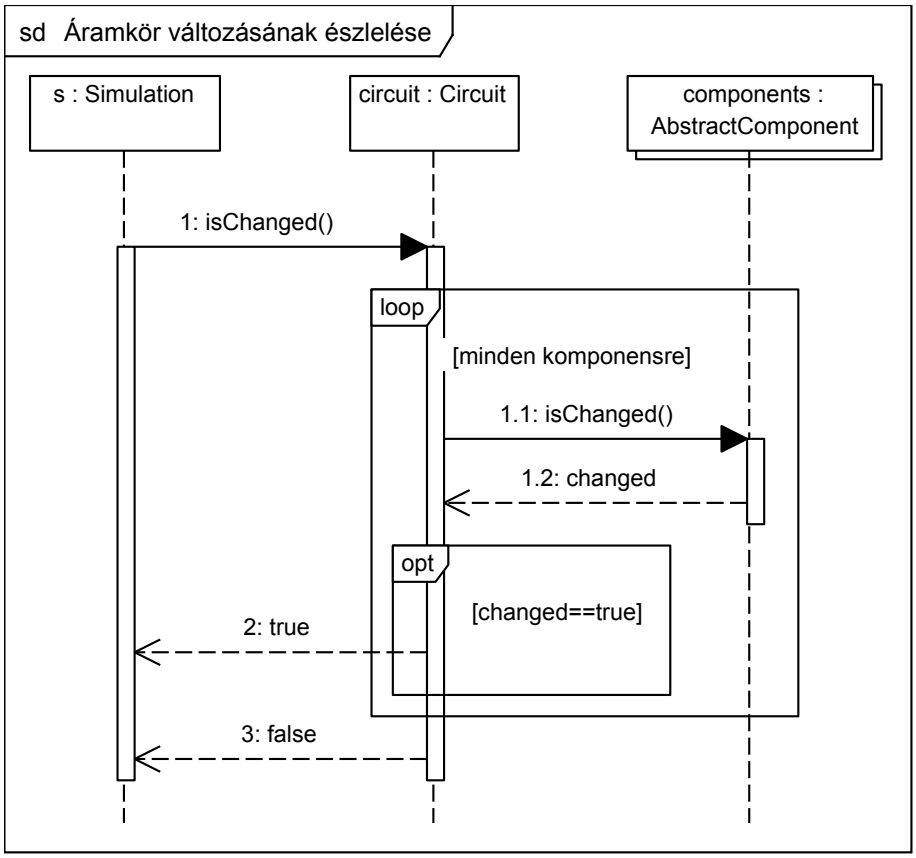
### 4.4. Szekvencia diagramok



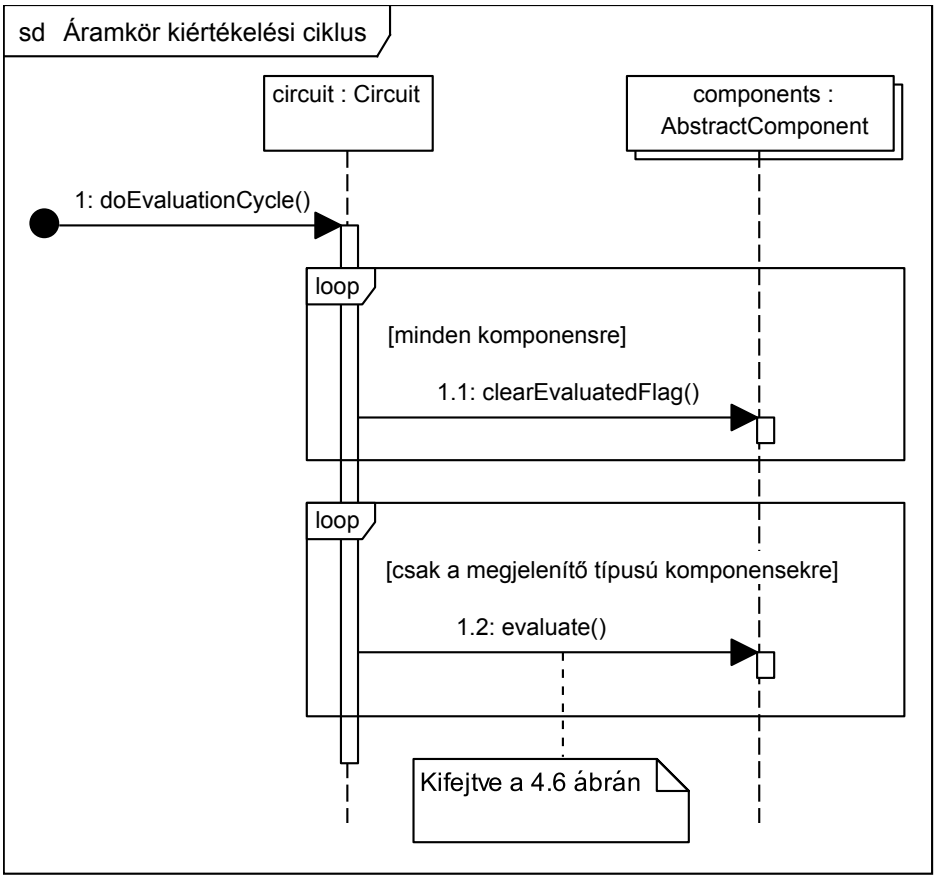

4.2. ábra. Inicializálás



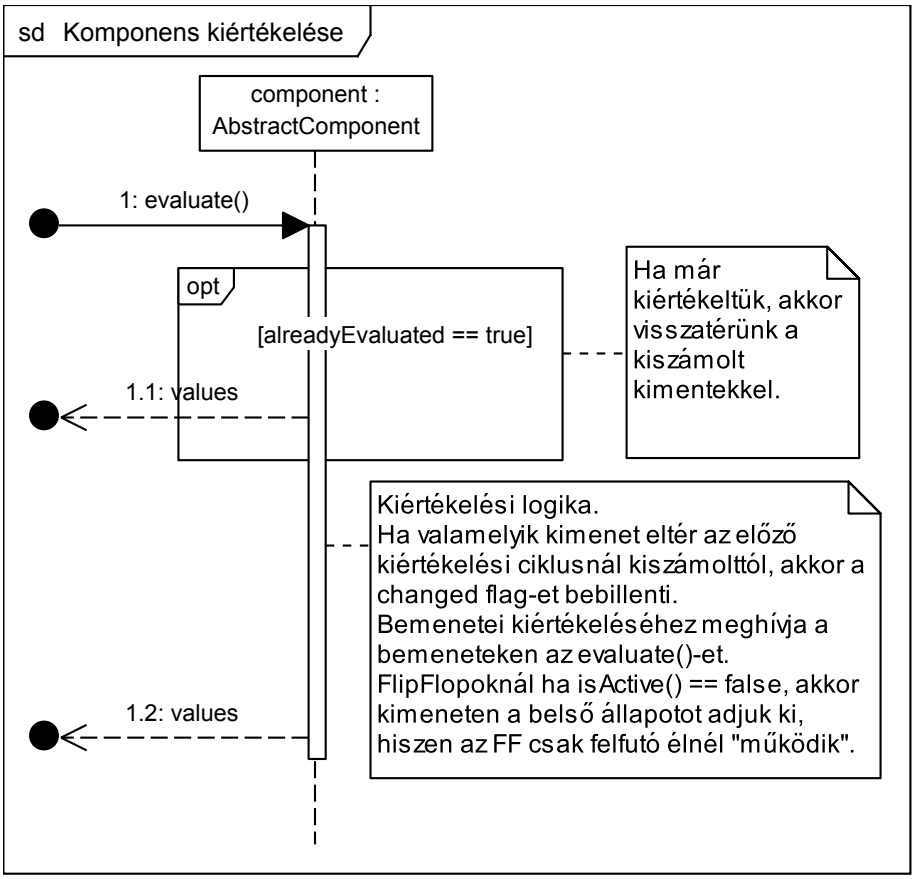

4.3. ábra. Szimuláció



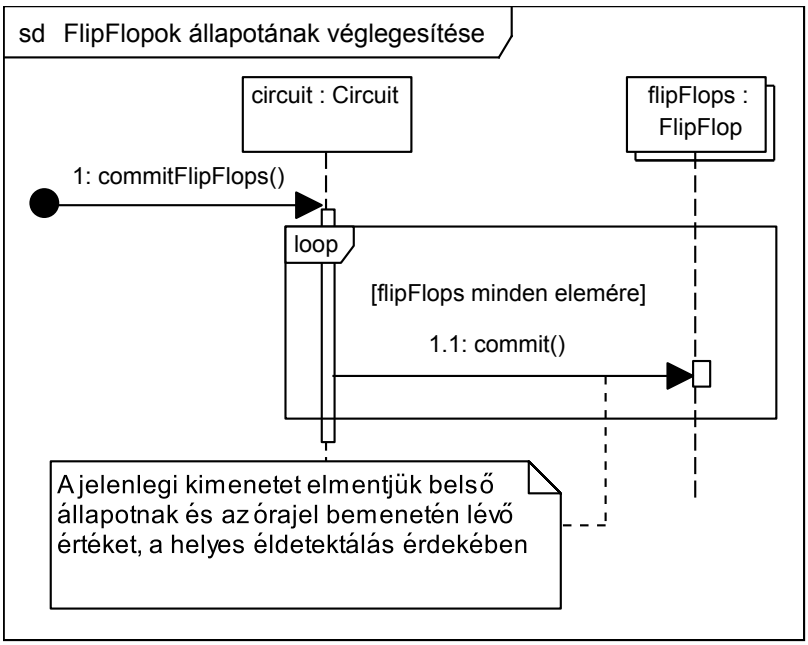

4.4. ábra. Áramkör változásának észlelése



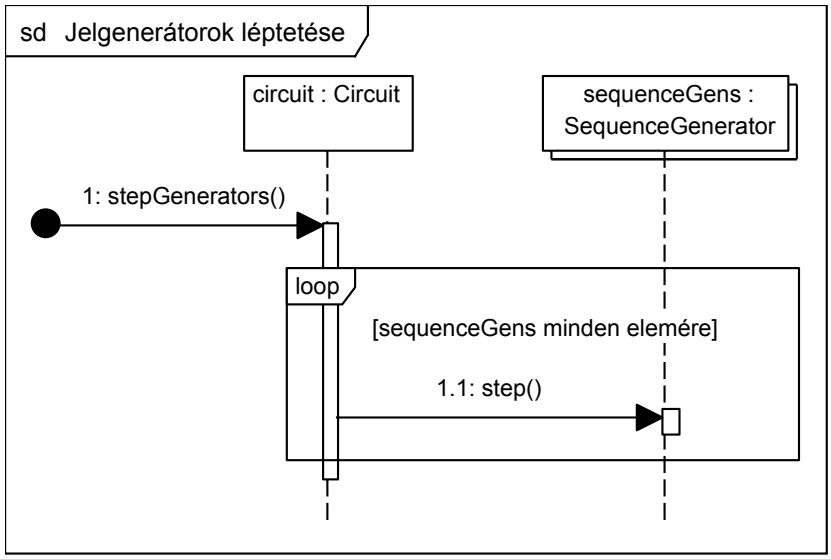

4.5. ábra. Áramkör kiértékelési ciklus



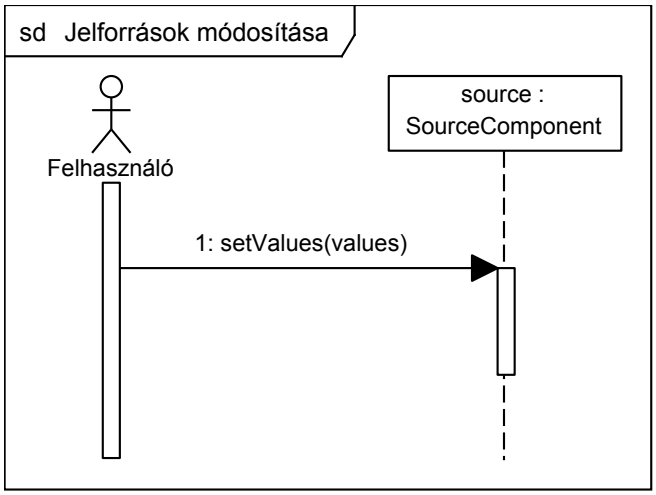

4.6. ábra. Komponens kiértékelése




4.7. ábra. Flipflopok állapotának véglegesítése

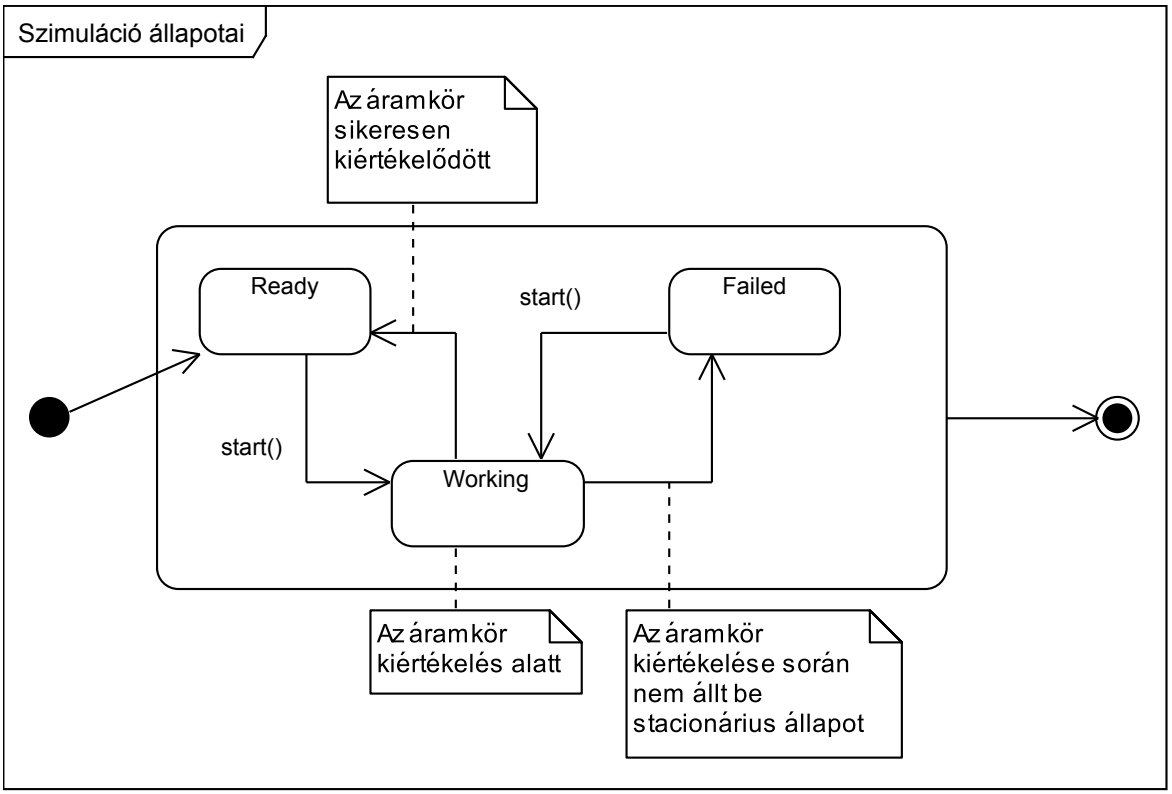



4.8. ábra. Jelgenerátorok léptetése




4.9. ábra. Jelforrások módosítása

### 4.5. State-chartok




4.10. ábra. Szimuláció állapotgépe

### 4.6. Napló

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2011.03.02. 9:00 | 1,5 óra | **Apagyi G.**<br>**Dévényi A.**<br>**Jákli G.**<br>**Kriván B.**<br>**Péter T.** | Konzultáción elhangzottak megtárgyalása, diagramok revíziója |
| 2011.03.02. 17:00 | 1 óra | **Jákli G.** | Új diagram: áramkör betöltése fájlból |
| 2011.03.02. 18:00 | 1 óra | **Jákli G.** | Új diagram: inicializálás |
| 2011.03.02. 18:00 | 2 óra | **Kriván B.** | Új diagram: szimuláció |
| 2011.03.03. 15:00 | 1 óra | **Jákli G.** | Új diagram: szimuláció állapotgépe |
| 2011.03.03. 18:00 | 1 óra | **Dévényi A.** | Új diagram: jelgenerátorok léptetése, jelforrások módosítása |
| 2011.03.03. 18:00 | 1 óra | **Kriván B.** | Új diagram: áramkör kiértékelési ciklus |
| 2011.03.03. 20:00 | 1 óra | **Jákli G.** | Új diagram: áramkör változásának észlelése |
| 2011.03.03. 20:00 | 1 óra | **Dévényi A.** | Osztálydiagram újraszerkesztése |
| 2011.03.04. 18:00 | 30 perc | **Apagyi G.**<br>**Kriván B.** | Osztályok leírásának kritikus részeinek átbeszélése |
| 2010.03.04. 18:30 | 2 óra | **Apagyi G.** | Objektum katalógus és osztályok leírásának átdolgozása az aktuális modellnek megfelelően |

| Kezdet | Időtartam | Résztvevők | Leírás |
|---|---|---|---|
| 2010.03.04. 20:00 | 30 perc | **Kriván B.**<br>**Dévényi A.** | Értekezlet.<br>Döntés: Komponensek áramkörbe való beregisztrálásának módosítása. (Visitor tervezési minta) |
| 2011.03.04. 21:00 | 30 perc | **Kriván B.** | Új diagram: flipflopok állapotának véglegesítése |
| 2010.03.05. 10:00 | 2 óra | **Kriván B.** | Komponensek áramkörbe való beregisztrálásának módosítása, osztálydiagram átalakítása, szekvenciadiagramok módosítása |
| 2010.03.05. 16:00 | 2 óra | **Kriván B.** | Objektum katalógus és osztályok leírásának átdolgozása az aktuális modellnek megfelelően, apróbb hibák javítása. |
| ... | ... | **...** | ... |