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 5.

Tartalomjegyzék

Ábrák jegyzéke

4. Analízis modell kidolgozása 2

4.1. Objektum katalógus

4.1.1. Simulation

Szimuláció objektum. A szimulációért felelős. Elindítja a jelgenerátor léptetőt, s utasítja az áramkört több kiértékelési ciklus lefuttatásához, amíg az áramkörben van változás. Ha a változás megadott lépésen belül nem áll meg, tájékoztatja a felhasználót, hogy nincs stacionárius állapot. Amikor leállítódik, a jelgenerátor-léptetőt is leállítja.

4.1.2. Circuit

Az áramkör objektum. Ezen objektum feladata a jelgenerátor léptető kérésére a jelgenerátorok léptetése, az áramkörben található komponensek utasítása arra, hogy töröljék a "már kiértékelve" flaget egy adott kiértékelési ciklus előtt, hogy ezáltal a ciklusban minden kimenet értéke frissülhessen. Továbbá feladata a kiértékelés elindítása az összes kijelzőre, mert a rendszer kiértékelése a kijelzők kiértékelésével kezdődik.

4.1.3. SequenceGenerator

Jelgenerátor, az áramkört felépítő egyik alapelem, kiértékelési kezdeményezés hatására az előre betáplált jelsorozatot soron következő elemét állítja be aktuális értékként, így azon komponensek melyek bemenetére a Jelgenerátor van kötve, elérik aktuális értékét. Bemenete nem komponens jellegű így nem kezel más komponenseket. Mikor az áramkör kéri tőle, hogy lépjen, akkor a bitsorozat következő elemére lép.

4.1.4. AndGate

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

4.1.5. OrGate

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

4.1.6. Inverter

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

4.1.7. Gnd

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

4.1.8. Vcc

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

4.1.9. **Led**

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

4.1.10. **Toggle**

Kapcsoló, az áramkört felépítő elem, felhasználói interakciót követően, az aktuális értékét lehet állítani. Komponens bemenete nincs, így nem kezel további komponenseket.

4.1.11. **Value**

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

4.1.12. **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.

4.1.13. **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.

4.1.14. **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.

4.1.15. **SevenSegmentDisplay**

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

4.2. Osztályok leírása

4.2.1. Circuit

- Felelősség
  Feladata a jelgenerátor léptető kérésére a jelgenerátorok léptetése, a feldolgozó által létrehozott komponensek felvétele az áramkörbe, illetve ezek utasítása arra, hogy töröljék a "már kiértékelve" flaget egy adott kiértékelési ciklus előtt, hogy ezáltal a ciklusban minden kimenet értéke frissülhessen. Továbbá feladata a kiértékelés elindítása az összes kijelzőre, mert a rendszer kiértékelése a kijelzők kiértékelésével kezdődik.
- Ősosztályok: Object → Circuit.
- Interfészek: (nincs)
- Attribútumok
  - `private HashMap componentMap` Komponenseket tartalmazó HashMap
  - `private List displays` Megjelenítő típusú komponensek

  - `private List sources` Jelforrás típusú komponensek

- Metódusok
  - `AbstractComponent addComponent(AbstractComponent component)`: Komponens hozzáadása az áramkörhöz.
  - `void doEvaluationCycle()`: Egy kiértékelési ciklus lefuttatása. Az áramkörtől ezután lekérdezhető, hogy stabil (nem változott semelyik komponens kimenete az utolsó futtatás óta) vagy instabil állapotban van-e.
  - `List getDisplays()`: Megjelenítő típusú komponeseket adja vissza.
  - `List getSources()`: Jelforrás típusú komponenseket adja vissza.
  - `void stepGenerators()`: Jelgenerátorok a szimuláció szemszögéből nézve, egyszerre történő léptetése.
  - `void init()`: Előre meghatározott áramköri modell felépítése, objektumok létrehozása, kapcsolatok beállítása

4.2.2. Simulation

- Felelősség
  Egy szimulációt reprezentáló objektum. Futásakor elindítja a jelgenerátor léptetőt, s utasítja az áramkört több kiértékelési ciklus lefuttatásához, amíg az áramkörben van változás. Ha a változás megadott lépésen belül nem áll meg, tájékoztatja a felhasználót, hogy nincs stacionárius állapot. Amikor leállítódik, a jelgenerátor-léptetőt is leállítja. A szál természetéből adódóan többet nem indítható el, új szimulációhoz új példányt kell létrehozni.
- Ősosztályok: Object $\rightarrow$ Thread $\rightarrow$ Simulation.
- Interfészek: (nincs)
- Attribútumok
  - `private Circuit circuit` Szimulált áramkör
  - `private State state` Szimuláció jelenlegi állapota
- Metódusok
  - `Circuit getCircuit()`: Szimulált áramkör lekérdezése
  - `void start()`: Szimuláció elindítása
  - `void getState(State state)`: Szimuláció állapotának lekérdezése

4.2.3. Simulation.State

- Felelősség
  Szimuláció állapotait írja le
- Ősosztályok: Object $\rightarrow$ Enum $\rightarrow$ Simulation.State.
- Interfészek: (nincs)
- Attribútumok
  - `public static final State PAUSED` Szimuláció szüneteltetve van, a következő jelforrás változásig.

  - `public static final State STOPPED` Szimuláció leállt, ahhoz, hogy bármi történjen az áramkörre újra kell indítani.
  - `public static final State WORKING` Szimuláció éppen dolgozik, egy konkrét jelforrás-kombinációt alkalmazva dolgoztatja az áramkört

- Metódusok
  - `static State valueOf(String name):`
  - `static State[] values():`

4.2.4. Value

- Felelősség
  Az áramkörben előfordulható értéket reprezentál.
- Ősosztályok: Object → Enum → Value.
- 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.

4.2.5. 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: Object → AbstractComponent.
- Interfészek: Component.
- Attribútumok
  - `private boolean changed` Azt mutatja meg, hogy változott-e a komponens értéke
  - `protected boolean alreadyEvaluated` "Kiértékelt" flag, ha ez be van billenve, akkor nem számolunk újra, csak visszaadjuk az előzőleg kiszámolt értéket.
  - `protected Value[] Values` Jelenlegi (számolás közben) érték, ezt csak rövid ideig tároljuk, ahhoz kell, hogy tudjuk változik-e a lastValue-hoz képest.
  - `protected int[] indices` Itt tároljuk, hogy melyik bemenetre, az adott komponens melyik kimenetét kötöttük.
  - `protected AbstractComponent[] inputs` Az adott bemenetekre kötött komponensek.
  - `protected String name` Komponens neve (változó neve, ahogy a leíróban azonosítjuk)
- Metódusok

- `void clearEvaluatedFlag()`: Töröljük a komponens "kiértékelt" flagjét.
- `Value evaluate()`: Lekérjük a 0. kimenetén lévő értéket.
- `String getName()`: Név lekérdezése
- `void setName(String name)`: Név beállítása (változó, amivel azonosítjuk)
- `void setInput(int inputSlot, AbstractComponent component)`: Shortcut a másik setInput()-hoz, outputPin = 0-val.
- `void setInput(int inputPin, AbstractComponent component, int outputPin)`: Beállítunk egy bemenetet.
- `boolean isChanged()`: Volt-e változás, vagy nem?

4.2.6. Component

Interfész.

- Felelősség
  Komponens interfész, ebből származik az IsDisplay és az IsSource interfész. Legalapvetőbb dolgokat írja le (minden komponensnek van neve és értékei).
- Ősosztályok: (nincs)
- Interfészek: (nincs)
- Metódusok
  - `String getName()`: Név lekérdezése.
  - `Value getValue()`: Értéke lekérdezése a 0. kimeneten.
  - `Value getValue(int idx)`: Érték lekérdezése az adott kimeneten.
  - `void setName(String name)`: Név beállítása.

4.2.7. FlipFlop

Absztrakt osztály.

- Felelősség
  Flipflopok ősosztálya, minden flipflop 0. bemenete az órajel!
- Ősosztályok: Object $\rightarrow$ AbstractComponent $\rightarrow$ FlipFlop.
- Interfészek: (nincs)
- Attribútumok
  - `private boolean active` Ebben tároljuk, hogy a FF számolhat-e vagy sem. (felfutó él)
  - `protected Value q` A flip-flopban tárolt érték
- Metódusok
  - `boolean getActive()`: Számolhat-e az FF? Ezt hívja meg az FF-ek onEvaluation() metódusa, mielőtt bármit is csinálnának.
  - `void setActive(boolean active)`: Felfutó élnél a SequenceGenerator-nak meg kell hívni ezt a hozzá kötött FlipFlopokra, egyéb esetben törölnie az active flaget. Így tudja az FF, hogy mikor kell ténylegesen számolnia.
  - `void setInput(int inputPin, AbstractComponent component, int outputPin)`: Ősosztály implementációjának meghívása, illetve ha egy SequenceGeneratort kötünk éppen a CLK bemenetre, akkor az addFlipFlop() metódus meghívása rajta.

4.2.8. IsDisplay

Interfész.

- Felelősség
  Megjelenítő típusú komponenst reprezentál. Ezt kell implementálnia a megjelenítőknek.
- Ősosztályok: (nincs).
- Interfészek: Component.
- Metódusok
  - (nincs)

4.2.9. IsSource

Interfész.

- Felelősség
  Jelforrás típusú komponenst reprezentál. Ezt kell implementálnia a jelforrásoknak.
- Ősosztályok: (nincs).
- Interfészek: Component.
- Metódusok
  - `public Value[] getValues()`: Lekérhetjük a jelforrás értékeit, hogy el tudjuk menteni.
  - `public void setValues(Value[] values)`: Beállítjuk a jelforrás értékét. Kapcsoló esetén csak 1 elemű tömb adható paraméterként!

4.2.10. AndGate

- Felelősség
  ÉS kapu, az áramkör egyik alapeleme. Bemeneteire kötött komponensek kiértékelését kezdeményezi, s a kapott értékek logikai ÉS kapcsolatát valósítja meg, amit a kimenetén kiad.
- Ősosztályok: Object $\rightarrow$ AbstractComponent $\rightarrow$ AndGate.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

4.2.11. FlipFlopD

- Felelősség
  D flipflop, mely felfutó órajelnél beírja a belső memóriába az adatbemeneten (D) lévő értéket.
- Ősosztályok: Object $\rightarrow$ AbstractComponent $\rightarrow$ FlipFlop $\rightarrow$ FlipFlopD.
- Interfészek: (nincs)
- Attribútumok

  - (nincs)

- Metódusok
  - (nincs)

4.2.12. FlipFlopJK

- Felelősség
  JK flipflop, mely a belső memóriáját a Követelmények résznél leírt módon a J és K bemenetektől függően változtatja.
- Ősosztályok: Object → AbstractComponent → FlipFlop → FlipFlopJK.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

4.2.13. Gnd

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

4.2.14. Inverter

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

4.2.15. Led

- Felelősség
  Egy LED-et reprezentál, mely világít, ha bemenetén igaz érték van. 3 féle színe lehet, ezeket a Color enumeráció határozza meg.
- Ősosztályok: Object → AbstractComponent → Led.
- Interfészek: IsDisplay.
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

4.2.16. Mpx

- Felelősség
  4-1-es multiplexer, melynek a bemeneti lábak sorrendje a következő: D0, D1, D2, D3, S0, S1. Ahol Dx az adatbemenetek, Sy a kiválasztóbemenetek. Kimenetén a kiválasztóbemenetektől függően valamelyik adatbemenet kerül kiadásra.
- Ősosztályok: Object → AbstractComponent → Mpx.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

4.2.17. OrGate

- Felelősség
  VAGY kapu, az áramkör egyik alapeleme. Bemeneteire kötött komponensek kiértékelését kezdeményezi, s a kapott értékek logikai VAGY kapcsolatát valósítja meg, amit a kimenetén kiad.
- Ősosztályok: Object → AbstractComponent → OrGate.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

4.2.18. SequenceGenerator

- Felelősség
  Jelgenerátort reprezentál, amely a beállított bitsorozatot adja ki. A SequenceGeneratorStepper feladata, hogy a step() metódust meghívja ezen osztály példányain. Azokat a FF-eket vezérli, melyek CLK bemenetére ez a komponens van kötve, vagyis ha éppen felfutó él jön, akkor ezeket engedélyezi különben nem.
- Ősosztályok: Object → AbstractComponent → SequenceGenerator.
- Interfészek: IsSource.
- Attribútumok
  - `private List ffList`: Azon FF-ek listája, melyekre ez a jelgenerátor van bekötve a CLK bemenetre.
  - `private int index`: Bitsorozat egy indexe, ez határozza meg, hogy éppen melyik értéket adja ki.
  - `private Value[] sequence`: Tárolt bitsorozat
- Metódusok
  - `void addFlipFlop(FlipFlop ff)`: A flipflop-ot feliratkoztatjuk a jelgenerátorhoz, így ha felfutó él lesz, akkor tudunk neki jelezni.
  - `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.

4.2.19. SevenSegmentDisplay

- Felelősség
  7-szegmenses kijelzőt reprezentál, melynek 7 bemenete vezérli a megfelelő szegmenseket, ezek világítanak, ha az adott bemenetre logikai igaz van kötve.
- Ősosztályok: Object → AbstractComponent → SevenSegmentDisplay.
- Interfészek: IsDisplay.
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

4.2.20. Toggle

- Felelősség
  Kapcsoló jelforrás, melyet a felhasználó szimuláció közben kapcsolgathat.
- Ősosztályok: Object → AbstractComponent → Toggle.
- Interfészek: IsSource.
- Attribútumok

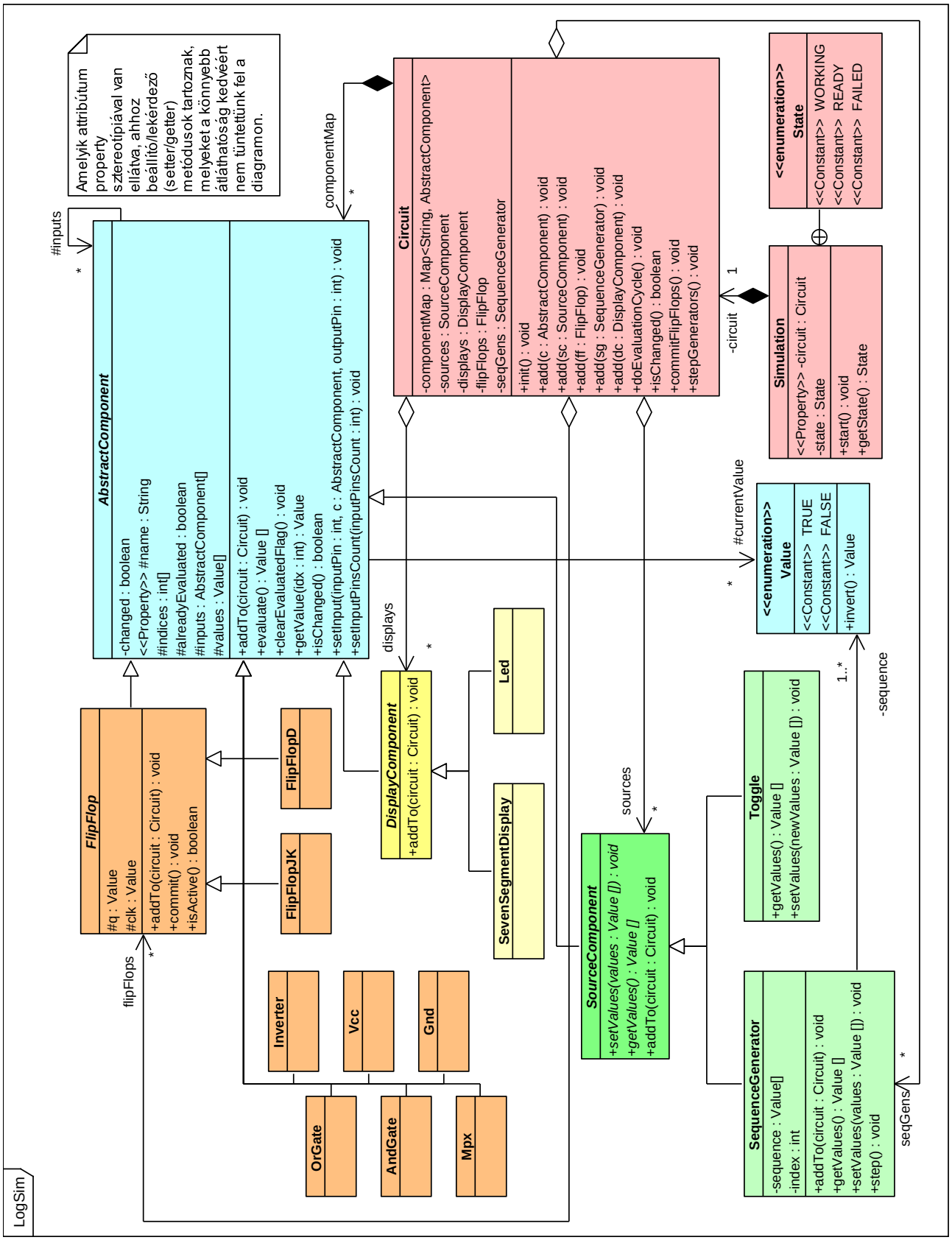
  - (nincs)

- Metódusok
  - `Value[] getValues()`: Lekérjük a kapcsoló értékét (1 elemű tömb)
  - `void setValues(Value[] values)`: Kapcsoló állapotának változtatása, csak 1 elemű tömböt kaphat paraméterül.

4.2.21. Vcc

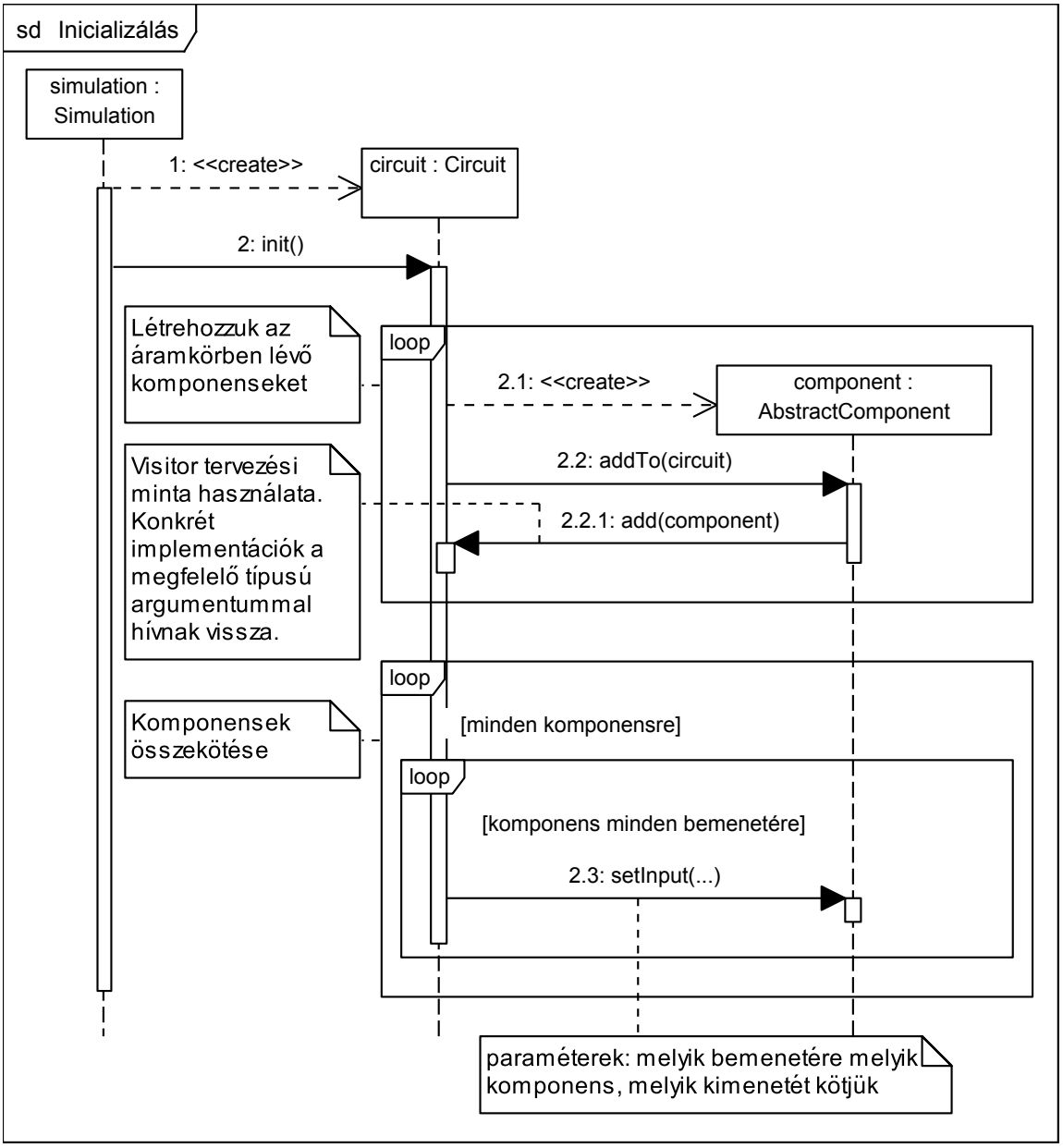
- Felelősség
  A tápfeszültés komponens, ami konstans igaz értéket ad. Nincs bemenete.
- Ősosztályok: Object $\rightarrow$ AbstractComponent $\rightarrow$ Vcc.
- Interfészek: (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

4.3. Statikus struktúra diagramok

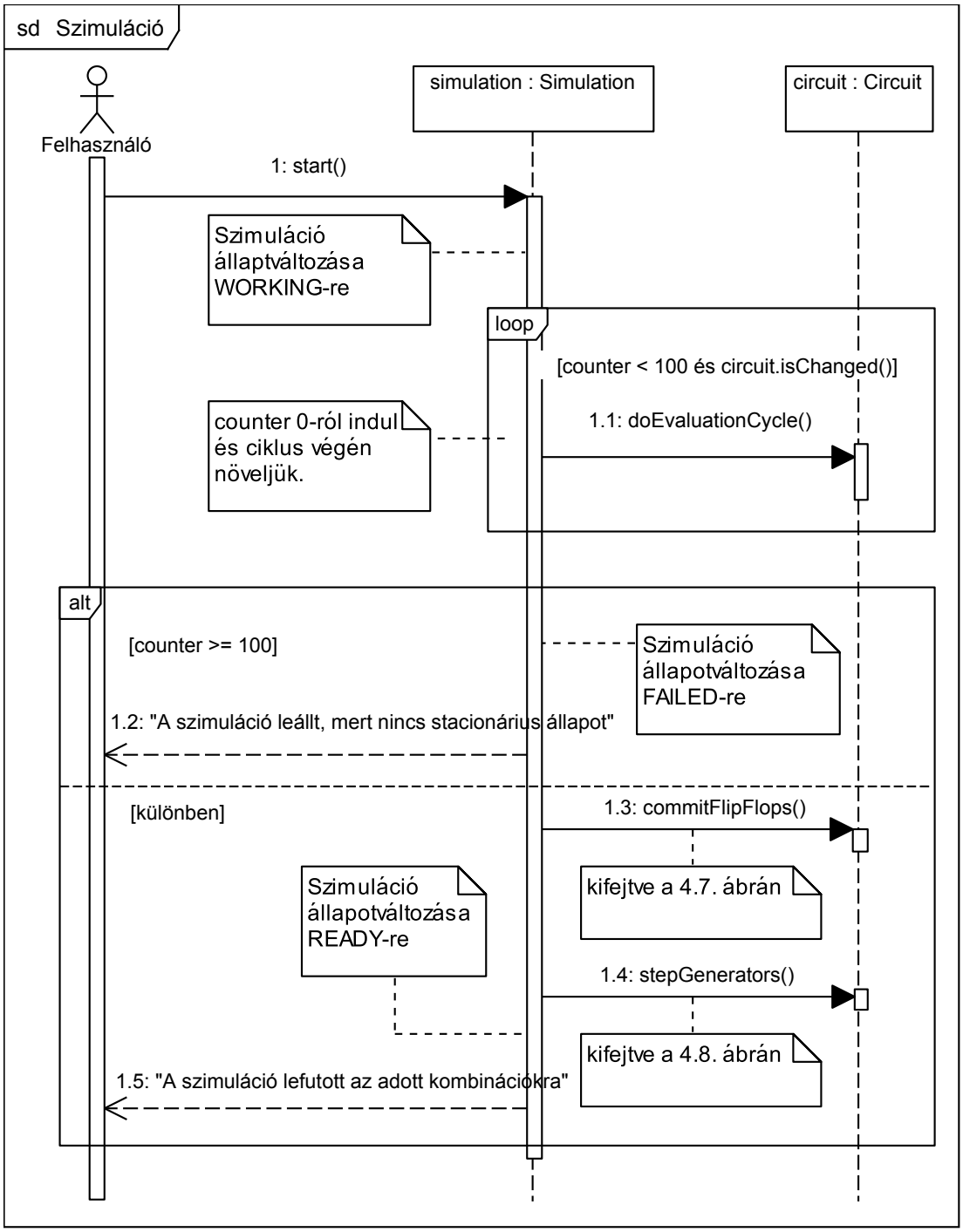


4.1. ábra. Statikus struktúra nézet

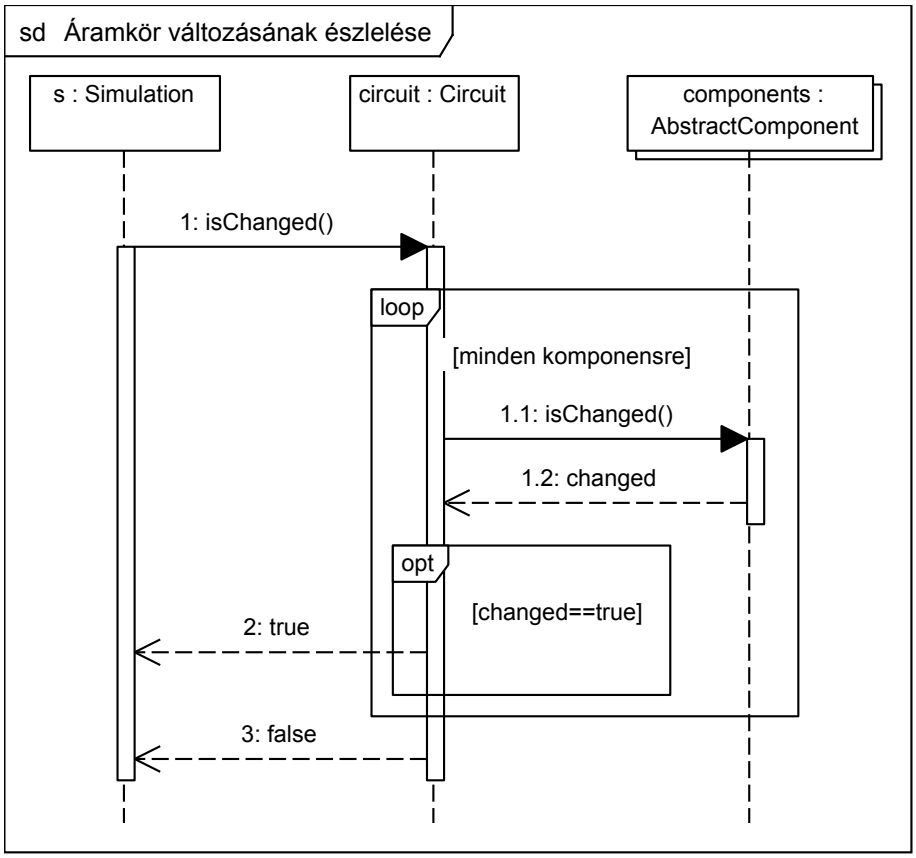
4.4. Szekvencia diagramok



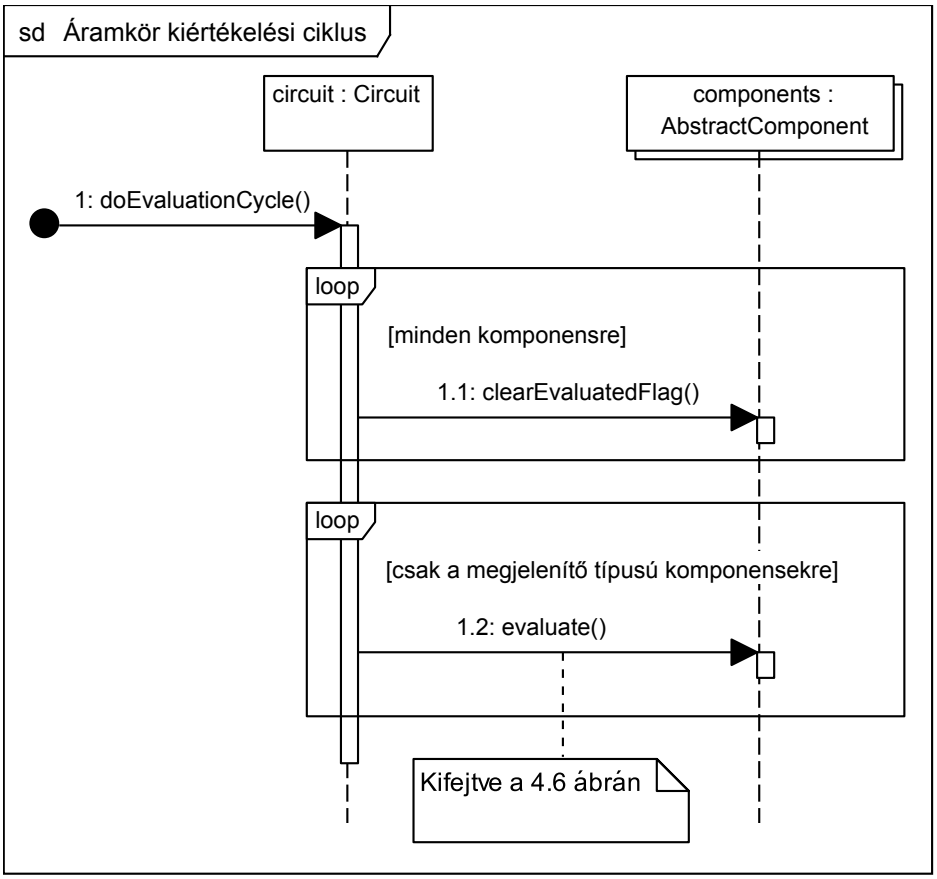
4.2. ábra. Inicializálás



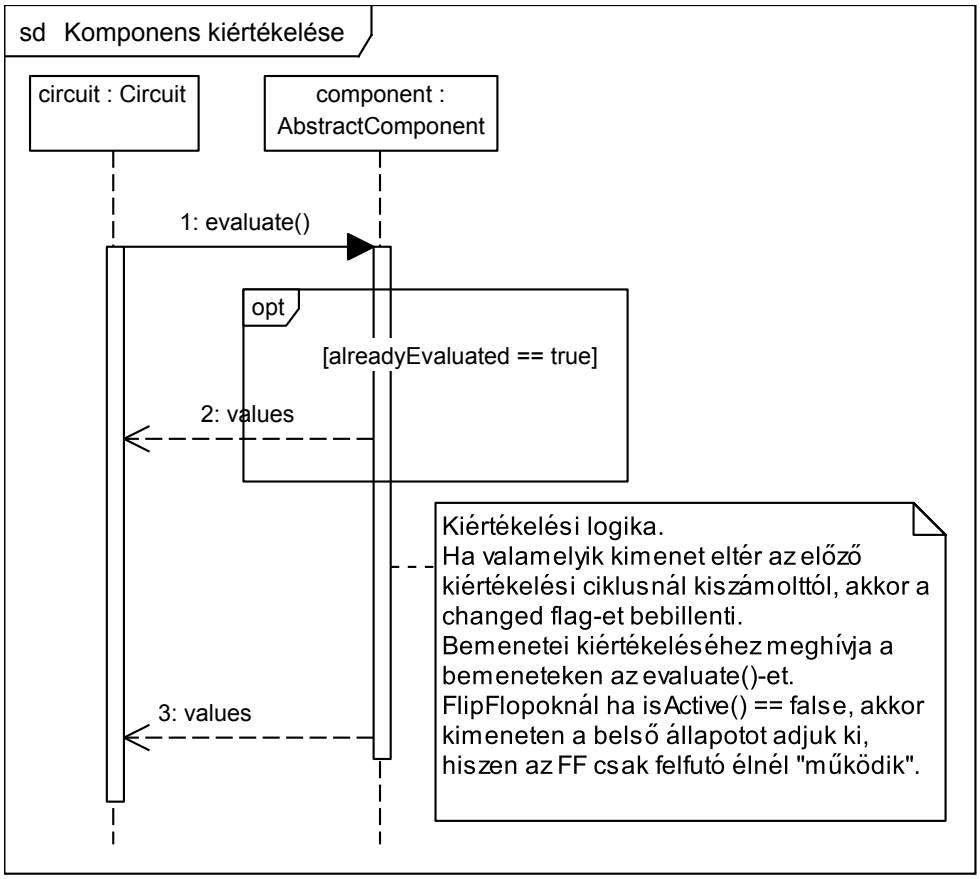
4.3. ábra. Szimuláció



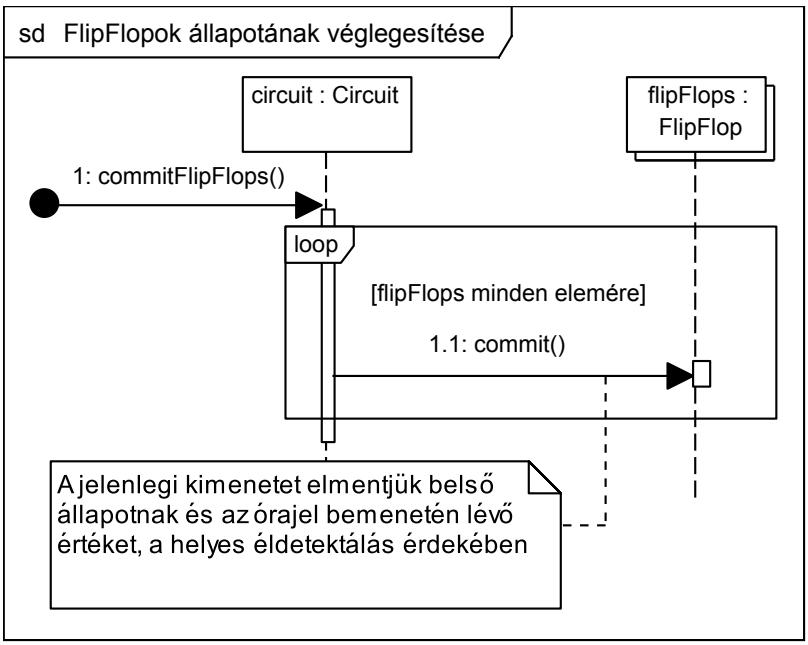
4.4. ábra. Áramkör változásának észlelése



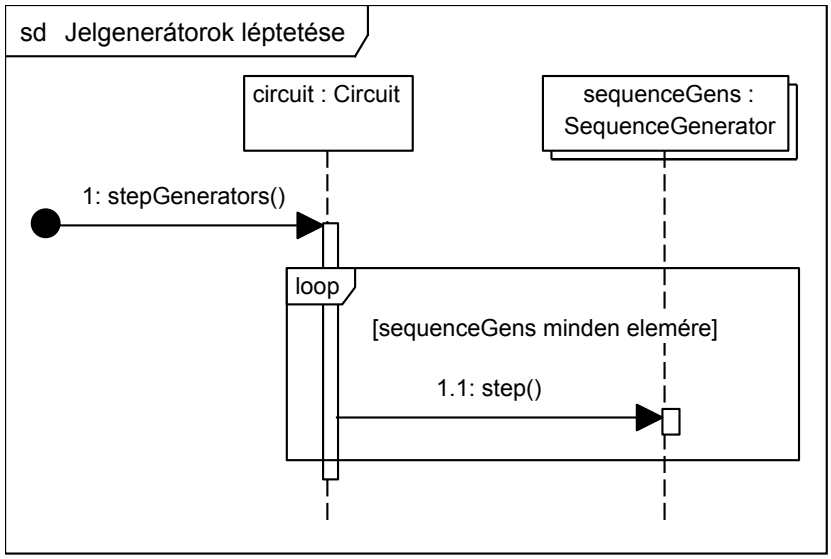
4.5. ábra. Áramkör kiértékelési ciklus



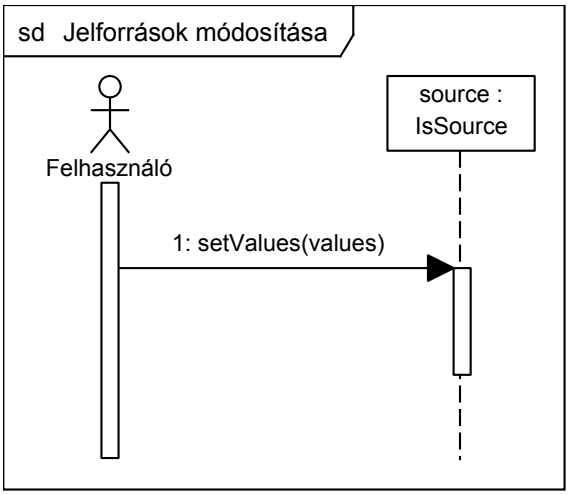
4.6. ábra. Komponens kiértékelése



4.7. ábra. Flipflopok állapotának véglegesítése

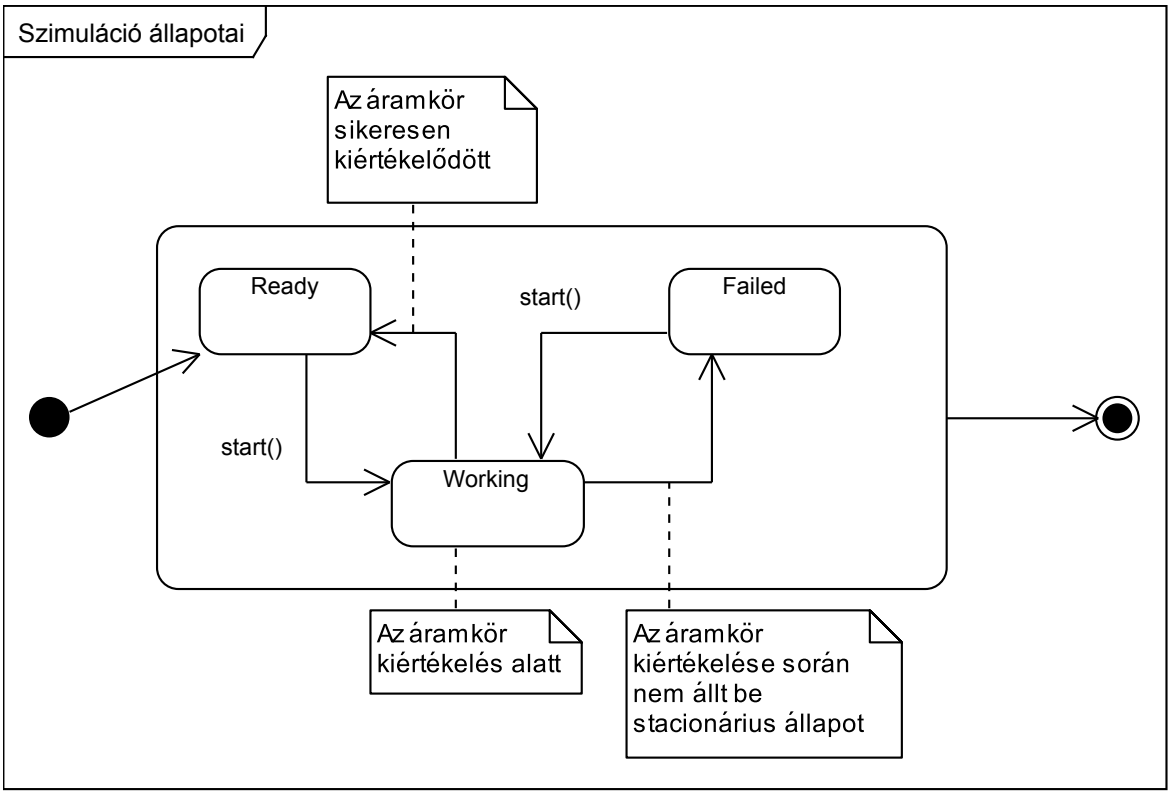


4.8. ábra. Jelgenerátorok léptetése



4.9. ábra. Jelforrások módosítása

4.5. State-chartok



4.10. ábra. Szimuláció állapotgépe

4.6. Napló

Kezdet	Időtartam	Résztvevők	Leírás
2011.03.04. 18:00	30 perc	**Apagyi G.** **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
...	...	**...**	...