3. Analízis modell kidolgozása 1

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. február 27.

Tartalomjegyzék

Ábrák jegyzéke

3. Analízis modell kidolgozása 1

3.1. Objektum katalógus

3.1.1. Parser

Áramkör értelmező objektum, feladata, hogy a paraméterként átadott, illetve fájlban elhelyezett komponenseket értelmezze, a kapcsolatokat feltérképezze, elvégezze az összeköttetéseket, és ezáltal felépítse az áramkört.

3.1.2. ConsoleView

Az áramkör karakteres megjelenítéséért, és a szimuláció során a változások megjelenítésének frissítéséért felelős objektum.

3.1.3. Simulation

Szimuláció objektum. A szimulációért felelős. Elindítja a jelgenerátor léptetőt, s utasítja az áramkört a kiértékelésre, és figyeli ha az áramkörben változás történt. Ha változás megadott lépésen belül nem történt, tájékoztatja a felhasználót, hogy nincs stacionárius állapot. Továbbá a megadott grafikai megjelenítőt frissíti.

3.1.4. 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, hogy ezáltal a következő kiértékelés kezdeményezésre továbbítsák azt bemeneteik számára is. Továbbá feladata a kiértékelés elindítása az összes kijelzőre, mert a rendszer kiértékelése a kijelzők kiértékelésével kezdődik.

3.1.5. SequenceGeneratorStepper

Jelgenerátor léptető objektum. Feladata, hogy a szimulációt utasítsa, hogy az áramkörben megtalálható jelgenerátorokat léptesse.

3.1.6. 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.

3.1.7. AndGate

ÉS kapu, az áramkör egyik alapeleme. Bemeneteire kötött komponensek kiértékelését kezdeményezi, s a kapott értékek logikai ÉS kapcsolatát valósítja meg, ezáltal a kimenetére kötött komponens eléri az aktuális értékét.Figyeli hogy ha már kiértékelődött akkor nem kezdeményezi a bemenetére kötött komponensek kiértékelését.

3.1.8. OrGate

VAGY kapu, az áramkör egyik alapeleme. Bemeneteire kötött komponensek kiértékelését kezdeményezi, s a kapott értékek logikai VAGY kapcsolatát valósítja meg, ezáltal a kimenetére kötött komponens eléri az aktuális értékét.Figyeli hogy ha már kiértékelődött akkor nem kezdeményezi a bemenetére kötött komponensek kiértékelését.

3.1.9. Inverter

Invertáló, az áramkör alapelemei közé tartozik. A bemenetére érkező jel logikai negáltját valósítja meg, így a kimenetén levő komponens eléri aktuális értékét.

3.1.10. Gnd

Föld, az áramkört felépítő egyik elem, aktuális értéke minden kiértékelési kérésre logikai hamis. Bemenete nem létezik, így nem kezdeményez további kiértékeléseket. Állandó értéke logikai hamis.

3.1.11. Vcc

Áramkör alapeleme, mely kiértékelési kezdeményezésre aktuális értékét logikai igaz ra állítja be. Állandó értéke logikai igaz.

3.1.12. Led

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

3.1.13. Toggle

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

3.2. Osztályok leírása

[Az előző alfejezetben tárgyalt objektumok felelősségének formalizálása attribútumokká, metódusokká. Csak publikus metódusok szerepelhetnek. Ebben az alfejezetben megjelennek az interfészek, az öröklés, az absztrakt osztályok. Segédosztályokra még mindig nincs szükség. Az osztályok ABC sorrendben kövessék egymást. Interfészek esetén az Interfészek, Attribútumok pontok kimaradnak.]

3.2.1. Circuit

- Felelősség
  Áramkört reprezentál, melyhez komponeseket lehet adni, és kiértékelési ciklusokat lehet futtatni, utóbbi a `Simulation` feladata.
- Ősosztályok Object → Circuit.
- Interfészek (nincs)
- Attribútumok
  - `private HashMap componentMap`
  - `private List sequenceGens`
  - `private Simulation simulation`

  - `private boolean stable`

- Metódusok
  - `public AbstractComponent addComponent(AbstractComponent component)`: Komponens hozzáadása az áramkörhöz.
  - `public 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.
  - `public AbstractComponent getComponentByName(String name)`: Lekérünk egy komponenst az áramkörtől a neve alapján.
  - `public List getDisplays()`: Megjelenítő típusú komponeseket adja vissza.
  - `public List getSequenceGenerators()`:
  - `public List getSources()`: Jelforrás típusú komponenseket adja vissza.
  - `public boolean isStable()`: Áramkör stacionárius állapotának lekérdezése.
  - `public void setSimulation(Simulation simulation)`: Szimuláció beállítása.
  - `public void setStable(boolean stable)`: Áramkör stabilitásának beállítása.
  - `public void simulationShouldBeWorking()`: Jelzi a szimuláció felé, hogy új ciklust kell indítani. Ezt egy jelforrás beállítása után hívjuk meg.
  - `public void stepGenerators()`: Jelgenerátorok a szimuláció szemszögéből nézve, egyszerre történő léptetése.

3.2.2. LogSim

- Felelősség

- Ősosztályok Object $\rightarrow$ LogSim.
- Interfészek Controller.
- Attribútumok
  - `Circuit circuit`
  - `Simulation simulation`
  - `View view`
- Metódusok
  - `public Simulation getFreshSimulation()`:
  - `public static void main(String[] args)`:
  - `public void onCircuitUpdate()`:
  - `public void onExit()`:
  - `public void onStart()`:
  - `public void onStop()`:

3.2.3. SequenceGeneratorStepper

- Felelősség

- Ősosztályok Object → Thread → SequenceGeneratorStepper.
- Interfészek (nincs)
- Attribútumok
  - `private long pause`
  - `private boolean shouldRun`
  - `private Simulation simulation`
- Metódusok
  - `public void run():`

3.2.4. Simulation

- Felelősség

- Ősosztályok Object → Thread → Simulation.
- Interfészek (nincs)
- Attribútumok
  - `private Circuit circuit`
  - `private final Controller controller`
  - `private AtomicInteger counter`
  - `private State currentState`
  - `private static final int cycleLimit`
  - `private final Object lock`
  - `private SequenceGeneratorStepper seqGenStepper`
  - `private boolean shouldRun`
  - `private final Object synchObj`
- Metódusok
  - `public Circuit getCircuit():`
  - `public Object getLock():`
  - `public void run():`
  - `public void saveSources(String fileName):`
  - `public void setCircuit(Circuit circuit):`
  - `public void setState(State state):`

3.2.5. Simulation.State

- Felelősség
  Szimuláció állapotait írja le
- Ősosztályok Object → Enum → 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 jelforráskombinációt alkalmazva dolgoztatja az áramkört
- Metódusok
  - `public static State valueOf(String name):`
  - `public static State[] values():`

3.2.6. Value

- Felelősség
  Az áramkörben előfordulható érték
- Ősosztályok Object → Enum → Value.
- Interfészek (nincs)
- Attribútumok
  - `public static final Value FALSE`
  - `public static final Value TRUE`
- Metódusok
  - `public Value invert():`
  - `public static Value valueOf(String name):`
  - `public static Value[] values():`

3.2.7. AbstractComponent

Absztrakt osztály.

- Felelősség
- Ősosztályok Object → AbstractComponent.
- Interfészek Component.
- Attribútumok
  - `protected boolean alreadyEvaluated`

  - `protected Circuit circuit`
  - `protected Value[] currentValue`
  - `protected int[] indices`
  - `protected AbstractComponent[] inputs`
  - `protected Value[] lastValue`
  - `protected String name`
- Metódusok
  - `public void clearEvaluatedFlag()`:
  - `public Value evaluate()`:
  - `public Value evaluate(int outputPin)`: Számolás:
  - `public String getName()`:
  - `public Value getValue()`:
  - `public Value getValue(int idx)`:
  - `public void setCircuit(Circuit parent)`:
  - `public void setInput(int inputSlot, AbstractComponent component)`:
  - `public void setInput(int inputPin, AbstractComponent component, int outputPin)`: Beállítunk egy bemenetet.
  - `public void setInputPinsCount(int inputPinsCount)`:
  - `public void setName(String name)`:

3.2.8. Component

Interfész.

- Felelősség
- Ősosztályok Component.
- Interfészek (nincs)
- Metódusok
  - `public String getName()`:
  - `public Value getValue()`:
  - `public Value getValue(int idx)`:
  - `public void setName(String name)`:

3.2.9. FlipFlop

Absztrakt osztály.

- Felelősség
- Ősosztályok Object → AbstractComponent → FlipFlop.
- Interfészek (nincs)

- Attribútumok
  - `private boolean active`
- Metódusok
  - `public boolean isActive()`:
  - `public 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.

3.2.10. IsDisplay

Interfész.

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

3.2.11. IsSource

Interfész.

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

3.2.12. AndGate

- Felelősség
- Ősosztályok Object $\rightarrow$ AbstractComponent $\rightarrow$ AndGate.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

3.2.13. FlipFlopD

- Felelősség
  D flipflop, mely felfutó órajelnél beírja a belső memóriába az adatbemeneten (D) lévő értéket.
- Ősosztályok Object → AbstractComponent → FlipFlop → FlipFlopD.
- Interfészek (nincs)
- Attribútumok
  - `private static final int CLK`
  - `private static final int D`
- Metódusok
  - (nincs)

3.2.14. 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
  - `private static final int CLK`
  - `private static final int J`
  - `private static final int K`
- Metódusok
  - (nincs)

3.2.15. Gnd

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

3.2.16. Inverter

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

3.2.17. Led

- Felelősség
  Egy LED-et reprezentál, mely világít, ha
- Ősosztályok Object → AbstractComponent → Led.
- Interfészek IsDisplay.
- Attribútumok
  - `private Color color`
- Metódusok
  - `public Color getColor()`: Lekérdezzük a LED színét
  - `public void setColor(Color color)`: Beállítjuk a LED színét

3.2.18. Led.Color

- Felelősség
  LED-színt reprezentáló enum
- Ősosztályok Object → Enum → Led.Color.
- Interfészek (nincs)
- Attribútumok
  - `public static final Color BLUE`
  - `public static final Color RED`
  - `public static final Color YELLOW`
- Metódusok
  - `public static Color valueOf(String name):`
  - `public static Color[] values():`

3.2.19. 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.
- Ősosztályok Object → AbstractComponent → Mpx.
- Interfészek (nincs)
- Attribútumok
  - `private static final int DATA0`
  - `private static final int DATA1`
  - `private static final int DATA2`
  - `private static final int DATA3`
  - `private static final int SEL0`
  - `private static final int SEL1`
- Metódusok
  - (nincs)

3.2.20. OrGate

- Felelősség
  Vagy kaput reprezentál, melynek akkor van igaz érték a kimenetén, ha legalább egy bemenetén van igaz érték.
- Ősosztályok Object → AbstractComponent → OrGate.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

3.2.21. SequenceGenerator

- Felelősség
  Jelgenerátort reprezentál, amely a beállított bitsorozatot adja ki. A SequenceGeneratorStepper feladata, hogy a step() metódust meghívja ezen osztály példányain.
- Ősosztályok Object → AbstractComponent → SequenceGenerator.
- Interfészek IsSource.
- Attribútumok
  - `private int idx`
  - `private Value[] sequence`
- Metódusok

  - `public Value[] getValues():`
  - `public void setValues(Value[] values):`
  - `public void step()`: A jelgenerátor lép, a bitsorozat következő elemére ugrik. A következő léptetésig ez kerül kiadásra a kimeneteken.

3.2.22. SevenSegmentDisplay

- Felelősség
  7-szegmenses kijelzőt reprezentál, melyen 7 bemenete vezérli a megfelelő szegmenseket.
- Ősosztályok Object → AbstractComponent → SevenSegmentDisplay.
- Interfészek IsDisplay.
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

3.2.23. Toggle

- Felelősség
  Kapcsoló jelforrás, melyet a felhasználó szimuláció közben kapcsolgathat.
- Ősosztályok Object → AbstractComponent → Toggle.
- Interfészek IsSource.
- Attribútumok
  - (nincs)
- Metódusok
  - `public Value[] getValues():`
  - `public void setValues(Value[] values):`
  - `public void toggle()`: Kapcsoló állapotát megváltoztatjuk

3.2.24. Vcc

- Felelősség
  A tápfeszültés komponens, ami konstans igaz értéket ad.
- Ősosztályok Object → AbstractComponent → Vcc.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - (nincs)

3.2.25. Parser

- Felelősség
- Ősosztályok Object → Parser.
- Interfészek (nincs)
- Attribútumok
  - `private static final HashMap availableComponents`
  - `private Circuit circuit`
  - `private static Pattern componentPattern`
  - `private int constComps`
  - `private static Pattern inputPattern`
  - `private HashMap inputs`
- Metódusok
  - `public Circuit parse(File file)`: Létrehoz egy áramkört a megadott fájlból
  - `public Circuit parse(String[] content)`: Létrehoz egy áramkört az argumentumokban megadott komponensekből

3.2.26. SourceWriter

- Felelősség
  Kiírja egy fájlba a jelforrásokat
- Ősosztályok Object → SourceWriter.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok
  - `public void add(IsSource source)`: Hozzáadjuk a fájlhoz az adott jelforrás beállítását
  - `public void close()`: Bezárjuk a fájlt.

3.2.27. Osztály1

- Felelősség
  [Mi az osztály felelőssége. Kb 1 bekezdés.]
- Ősosztályok
  [Mely osztályokból származik (öröklési hierarchia)
  Legősebb osztály → Ősosztály2 → Ősosztály3...]
- Interfészek
  [Mely interfészeket valósítja meg.]
- Attribútumok
  [Milyen attribútumai vannak]

  - attribútum1: attribútum jellemzése: mire való
  - attribútum2: attribútum jellemzése: mire való

- Metódusok
  [Milyen publikus metódusokkal rendelkezik. Metódusonként egy-három mondat arról, hogy a metódus mit csinál.]
  - int foo(Osztály3 o1, Osztály4 o2): metódus leírása
  - int bar(Osztály5 o1): metódus leírása

3.2.28. Osztály2

- Felelősség
  [Mi az osztály felelőssége. Kb 1 bekezdés.]
- Ősosztályok
  [Mely osztályokból származik (öröklési hierarchia)
  Legősebb osztály → Ősosztály2 → Ősosztály3...]
- Interfészek
  [Mely interfészeket valósítja meg.]
- Attribútumok
  [Milyen attribútumai vannak]
  - attribútum1: attribútum jellemzése: mire való
  - attribútum2: attribútum jellemzése: mire való
- Metódusok
  [Milyen publikus metódusokkal rendelkezik. Metódusonként egy-három mondat arról, hogy a metódus mit csinál.]
  - int foo(Osztály3 o1, Osztály4 o2): metódus leírása
  - int bar(Osztály5 o1): metódus leírása

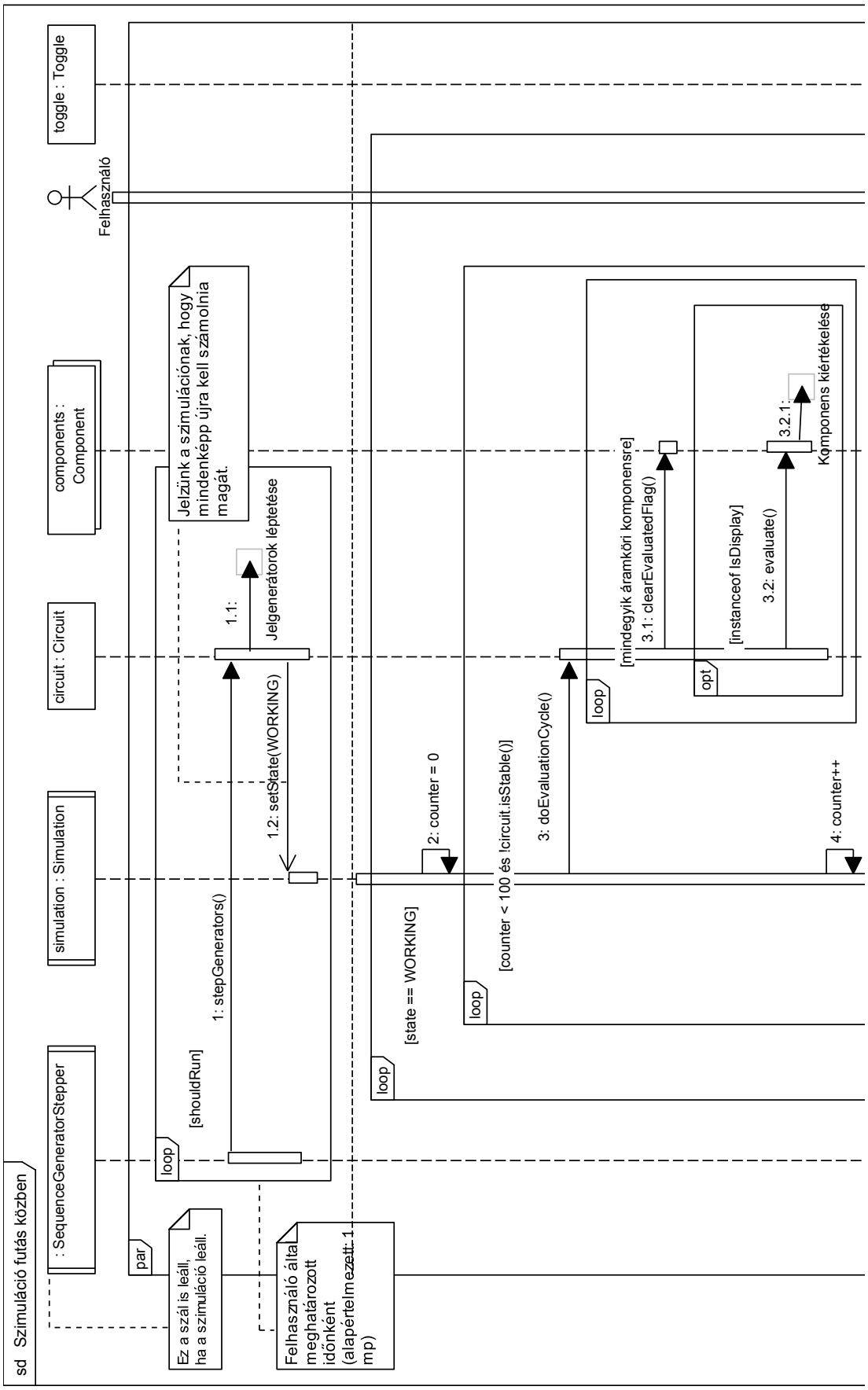
3.3. Statikus struktúra diagramok

[Az előző alfejezet osztályainak kapcsolatait és publikus metódusait bemutató osztálydiagram(ok). Tipikus hibalehetőségek: csillag-topológia, szigetek.]

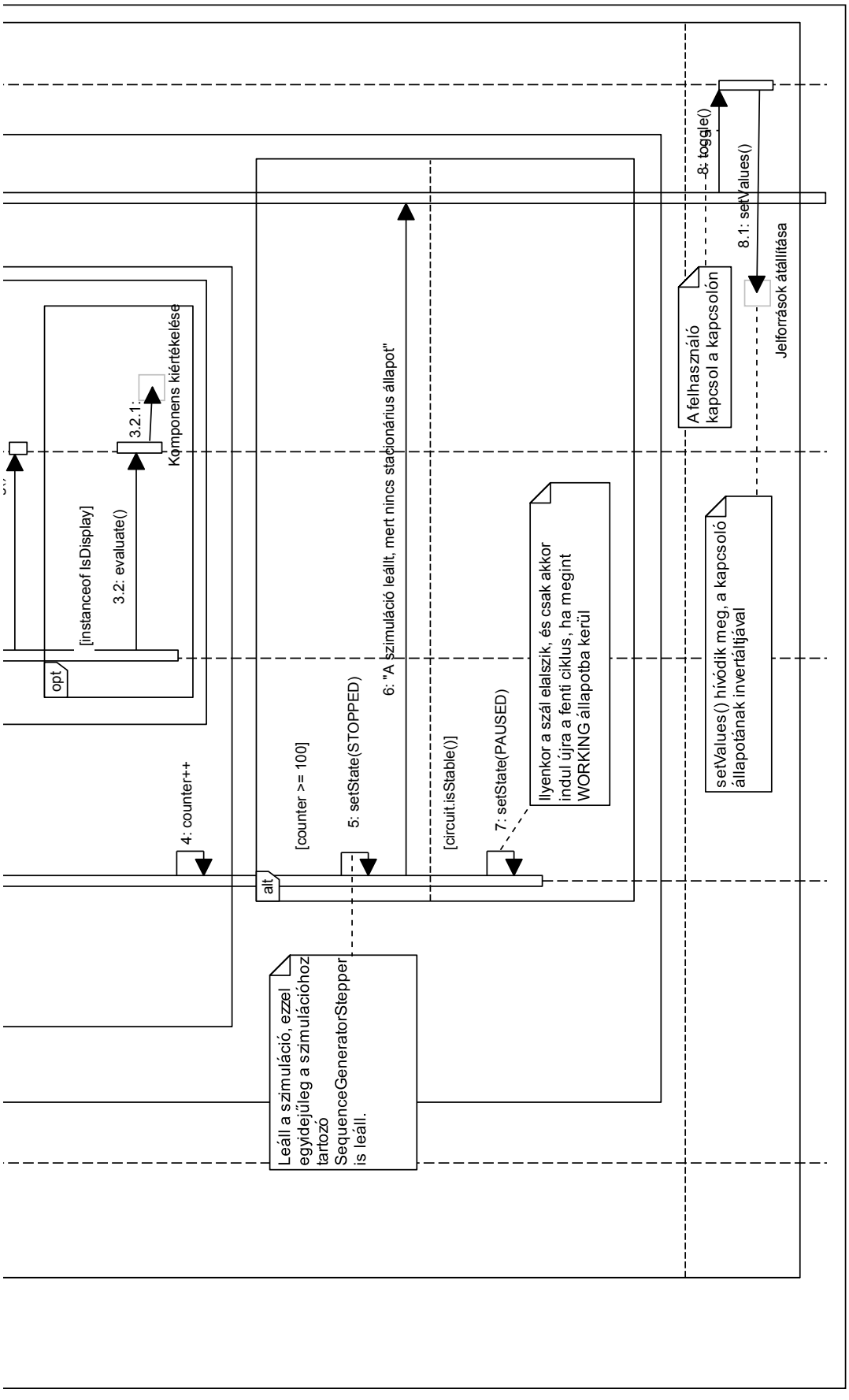
3.1. ábra. x

3.4. Szekvencia diagramok

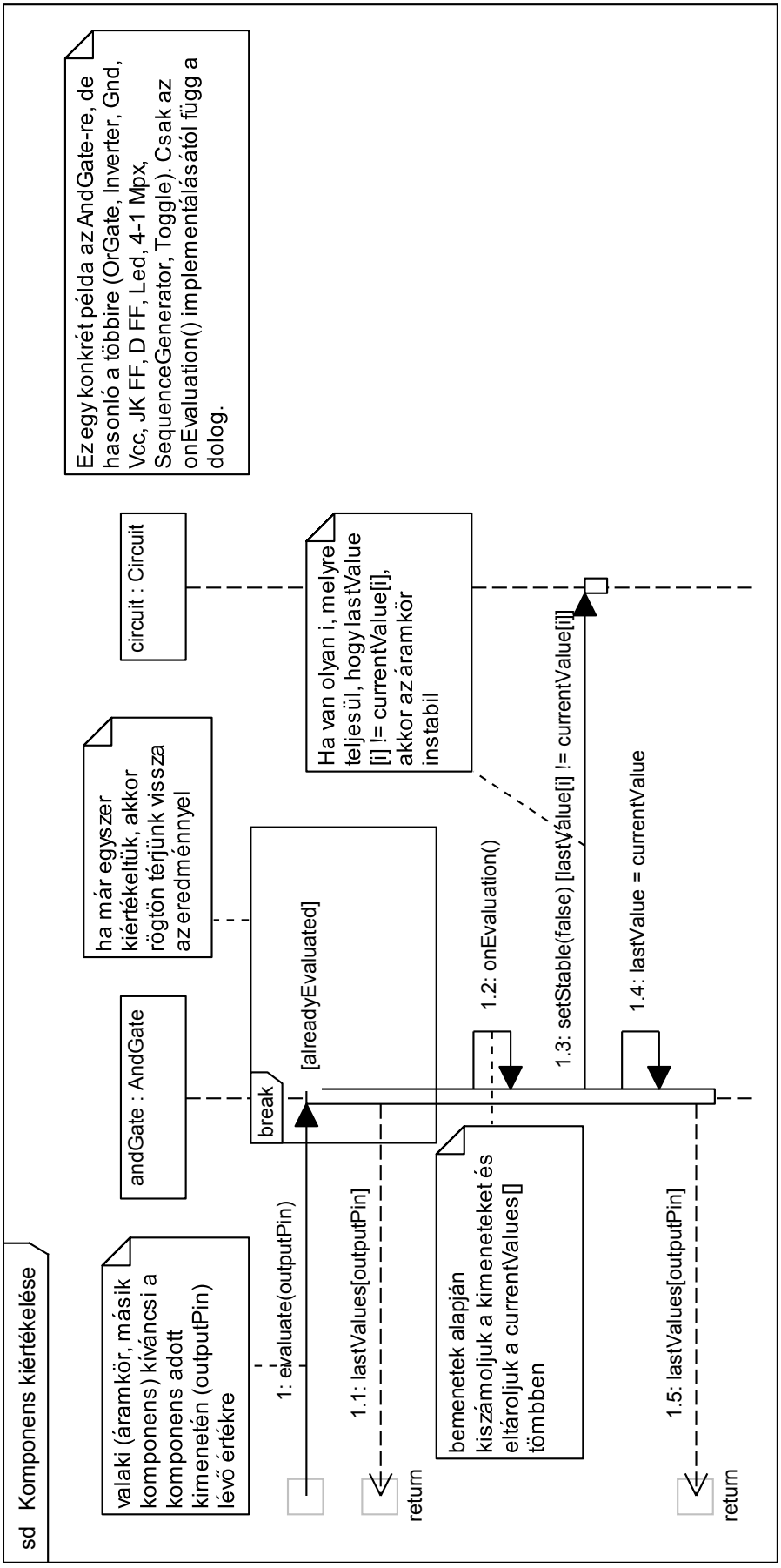
[Inicializálásra, use-case-ekre, belső működésre. Konzisztens kell legyen az előző alfejezettel. Minden metódus, ami ott szerepel, fel kell tűnjön valamelyik szekvenciában. Minden metódusnak, ami szekvenciában szerepel, szereplnie kell a valamelyik osztálydiagramon.]



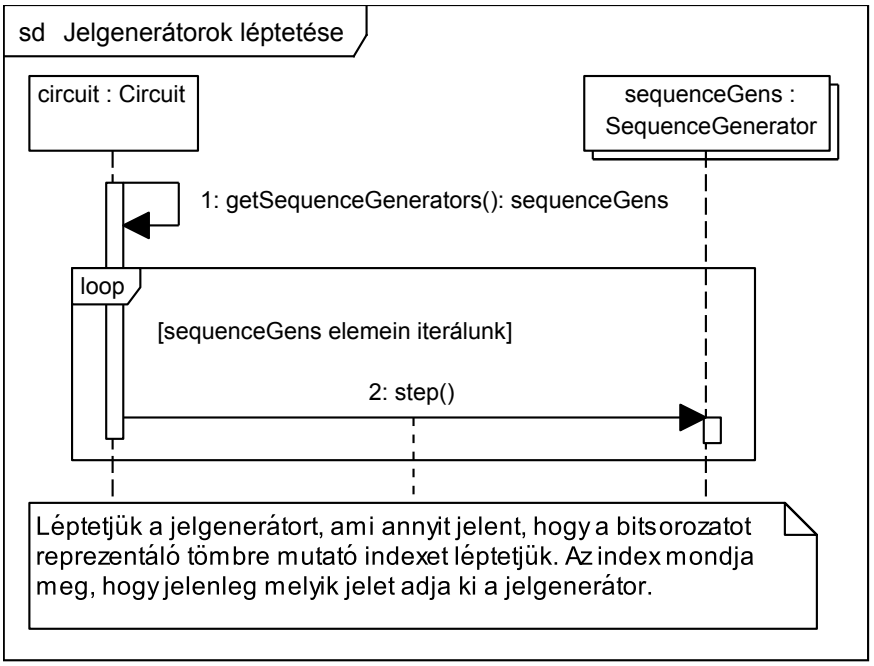
3.2. ábra. Szimuláció futás közben 1. rész (átfedéssel)



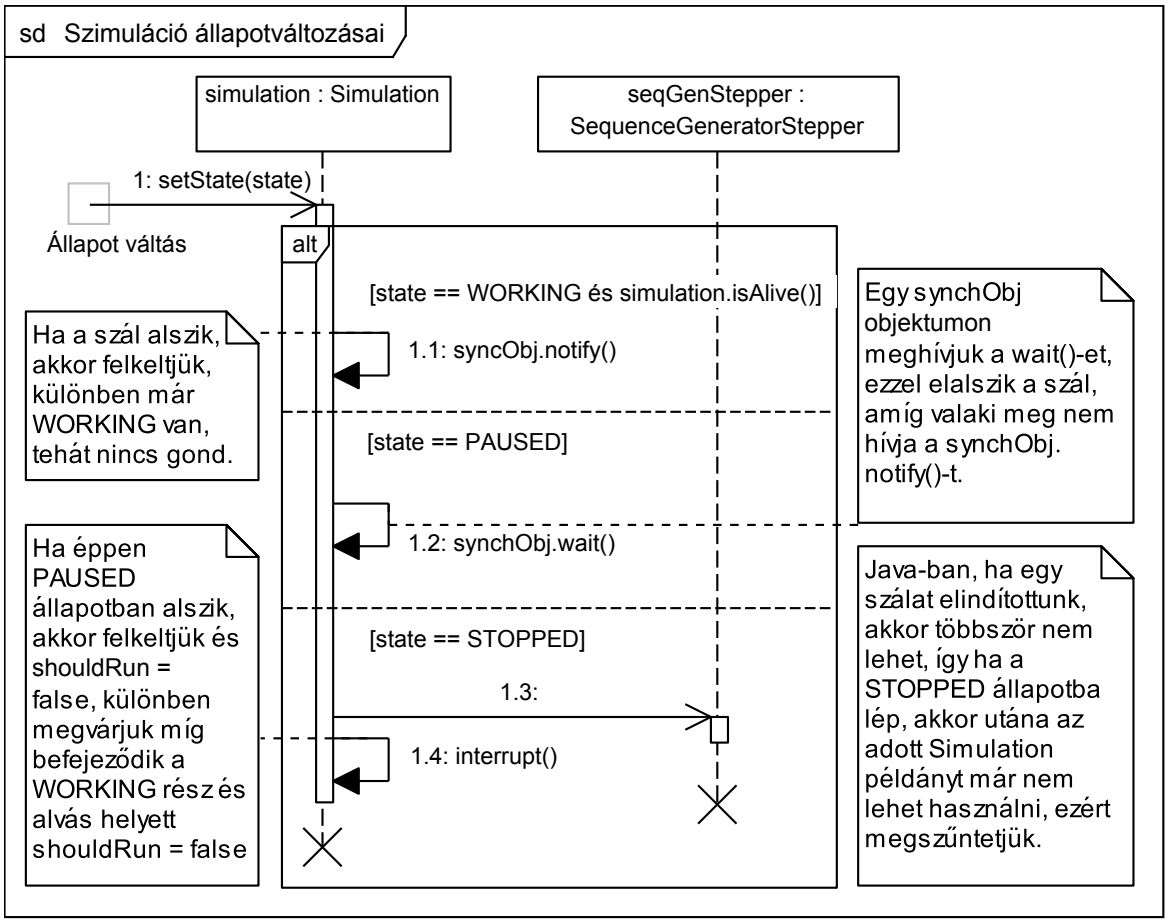
3.3. ábra. Szimuláció futás közben 2. rész



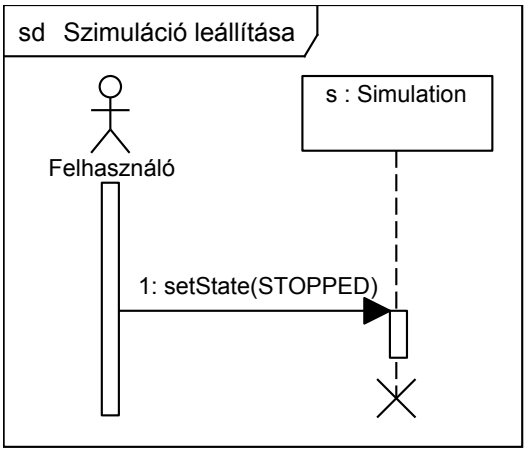
3.4. ábra. Komponens kiértékelése



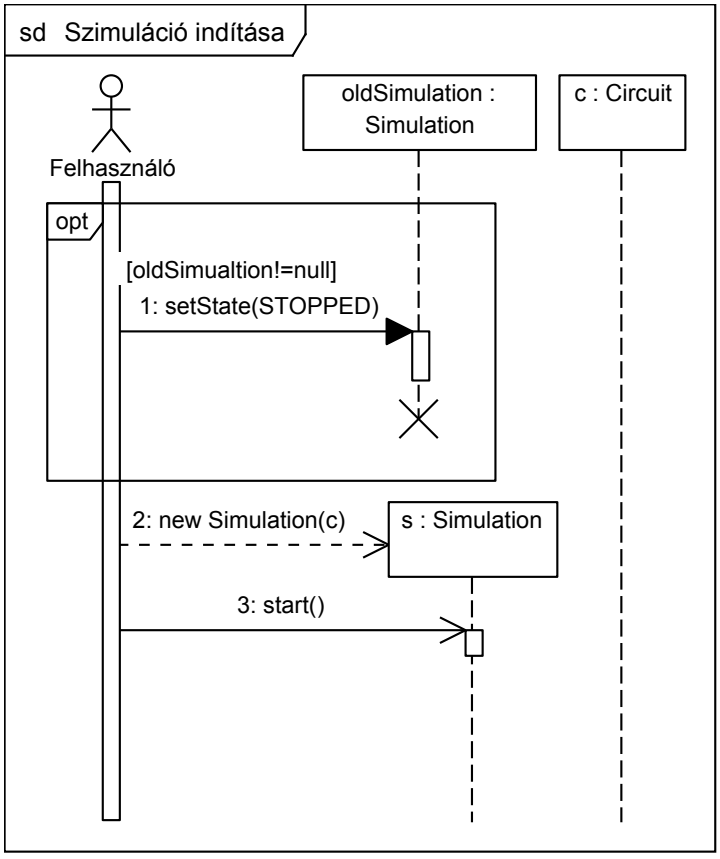
3.5. ábra. Jelgenerátorok léptetése



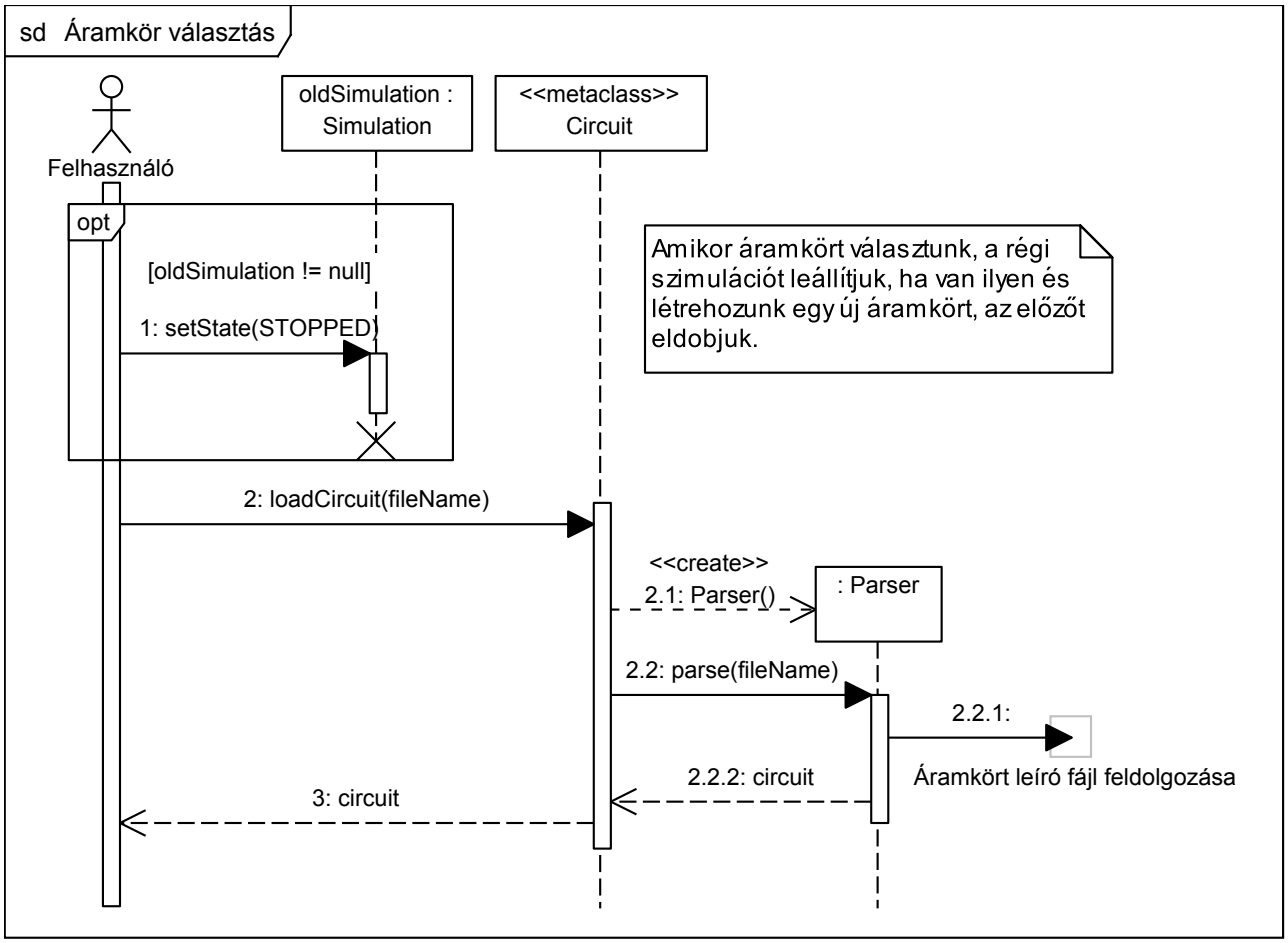
3.6. ábra. Szimuláció állapotváltozásai



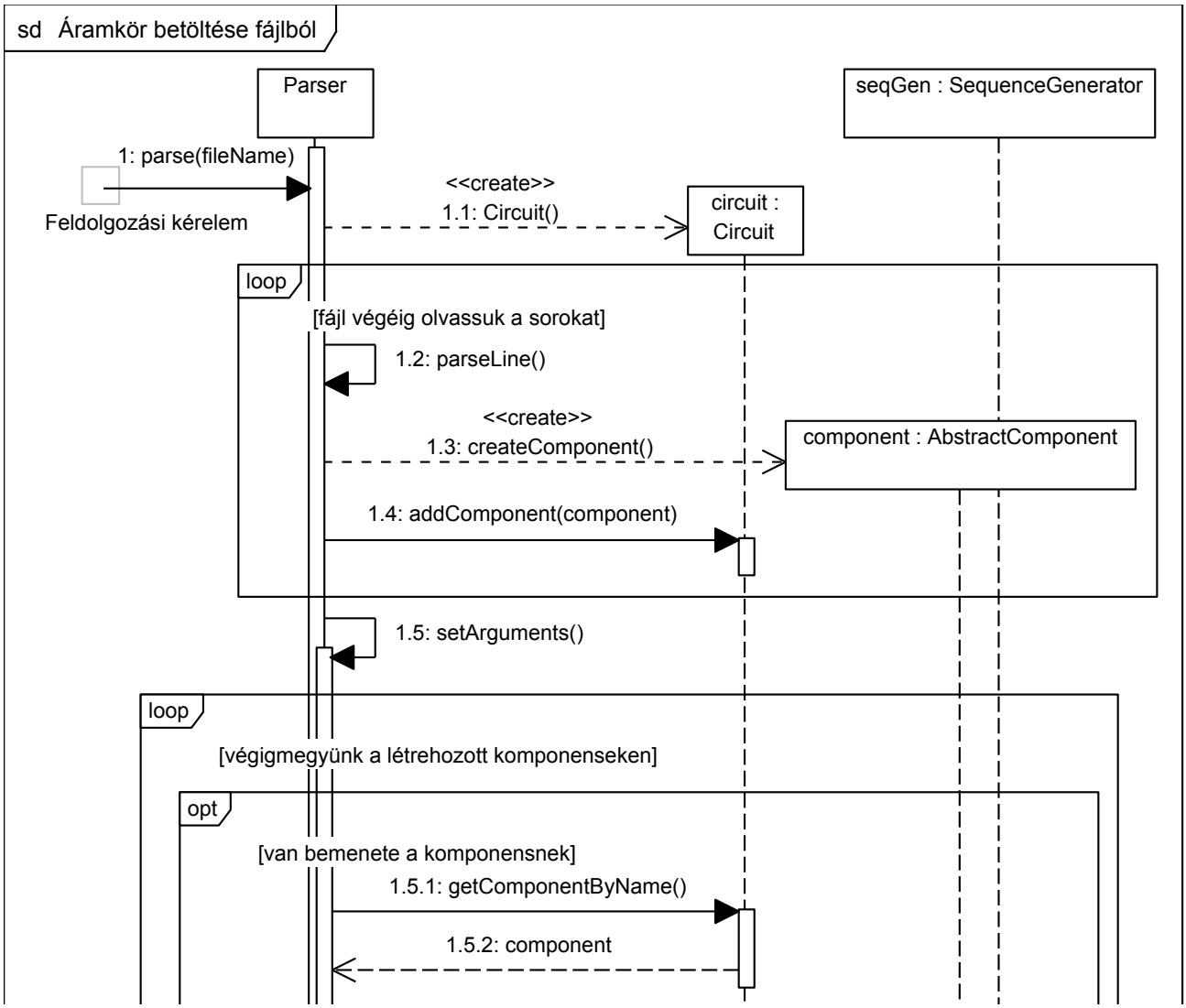
3.7. ábra. Szimuláció leállítása



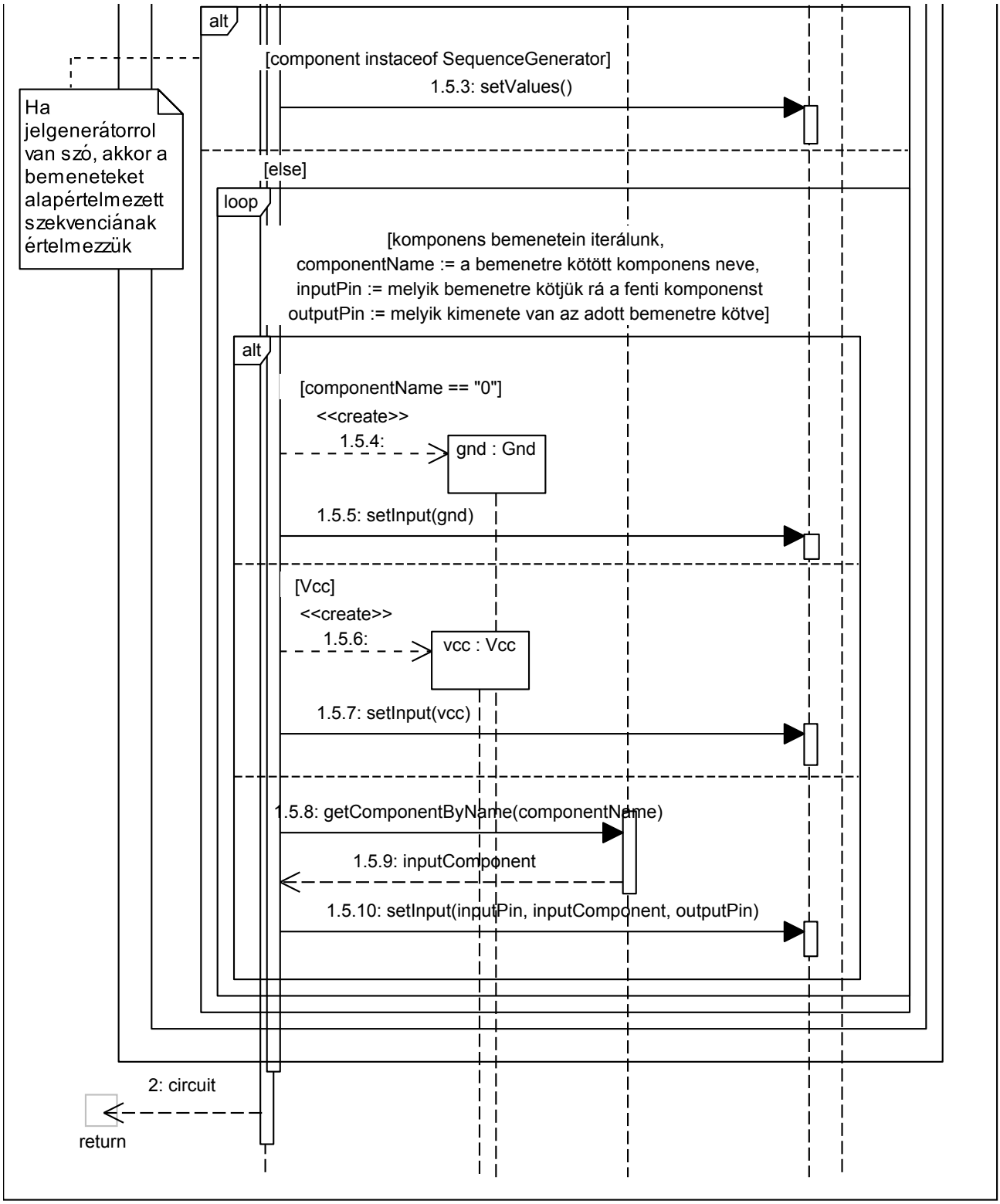
3.8. ábra. Szimuláció indítása



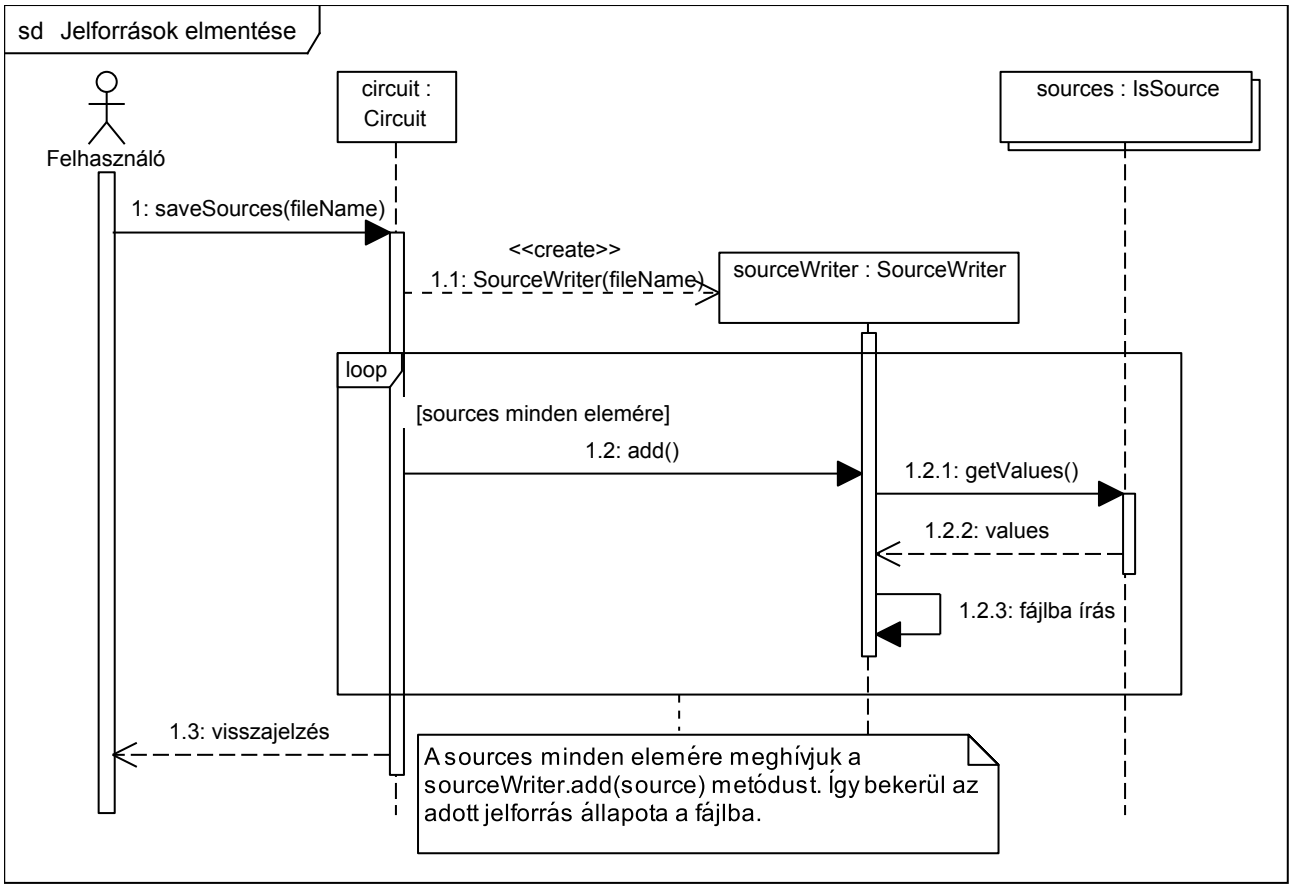
3.9. ábra. Áramkör választás



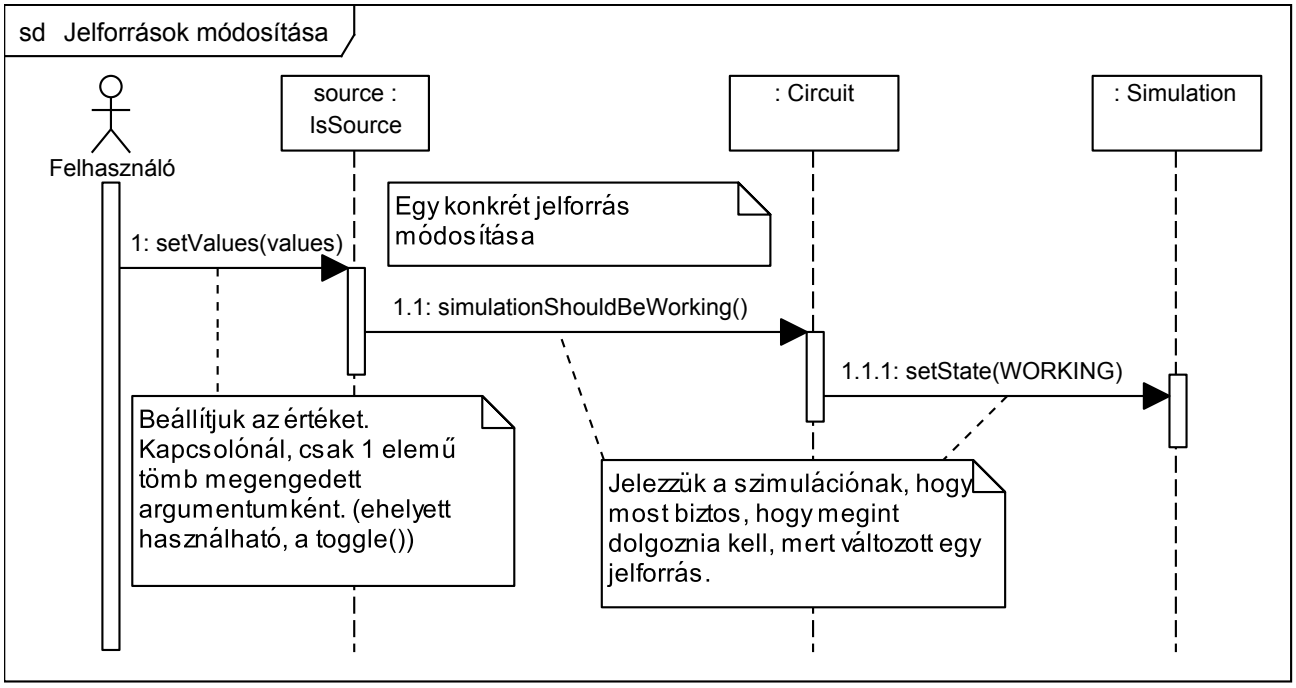
3.10. ábra. Áramkör betöltése fájlból 1. rész (vágva)



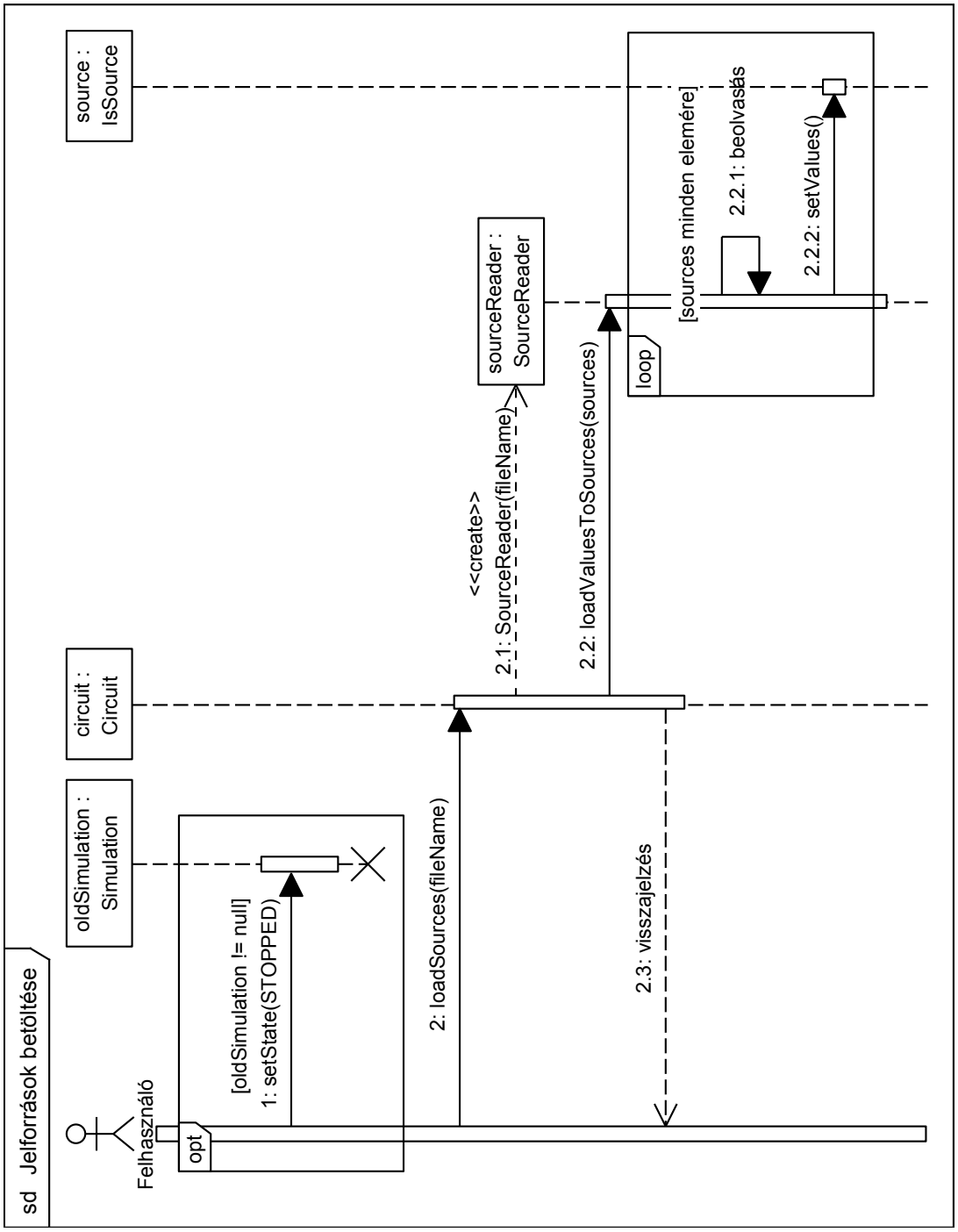
3.11. ábra. Áramkör betöltése fájlból 2. rész



3.12. ábra. Jelforrások mentése



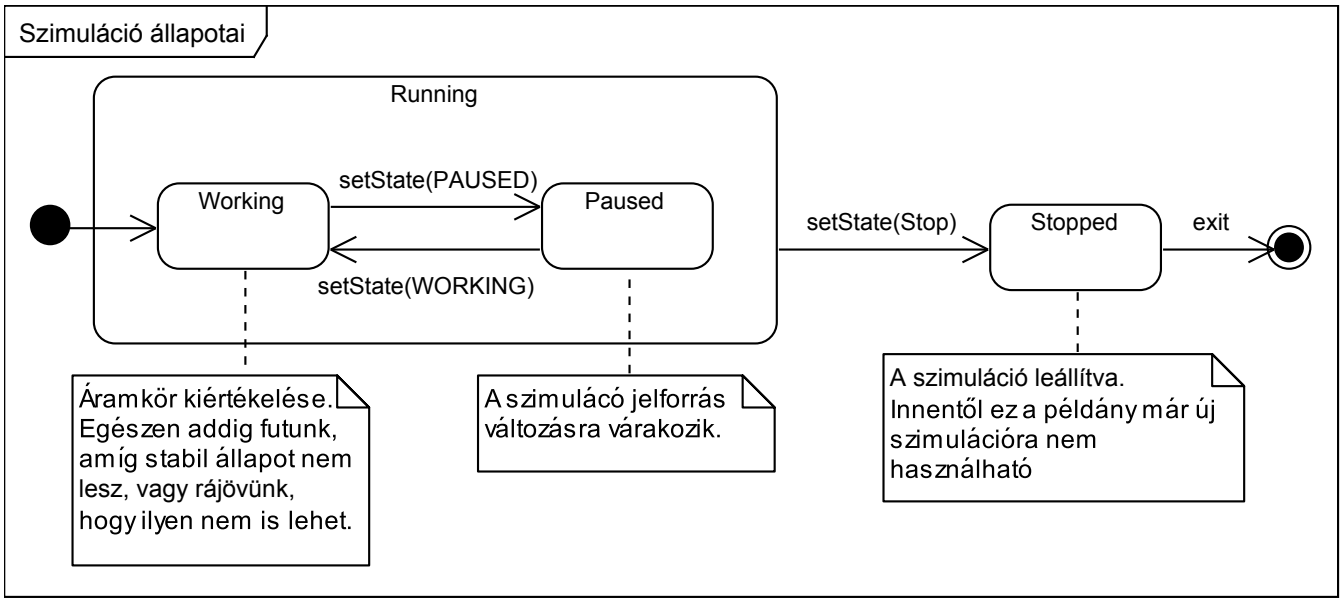
3.13. ábra. Jelforrások módosítása



3.14. ábra. Jelforrások betöltése

3.5. State-chartok

3.6. Napló



3.15. ábra. Szimuláció állapotai

Kezdet	**Időtartam**	**Résztvevők**	**Leírás**
2011.02.23. 10:00	30 perc	**Kriván B.** **Dévényi A.** **Péter T.** **Apagyi G.** **Jákli G.**	Analízis modell kezdeti lépéseinek megbeszélése
2011.02.23. 15:00	30 perc	**Péter T.**	LaTex és Visual Paradigm for UML szoftverek beállítása.
2011.02.25. 22:00	2 óra	**Péter T.**	Objektum katalógus kitöltése.
2011.02.26. 15:00	1 óra	**Apagyi G.** **Péter T.**	Objektum katalógus elemeinek átbeszélése
2011.02.26. 16:00	1 óra	**Apagyi G.**	Átállás a megbeszélt programokra (Latex, VP)
2011.02.26. 17:00	2 óra	**Apagyi G.**	Sequence diagram (start/stop) szerkesztése
2011.02.26. 18:00	2 óra	**Apagyi G.**	FF és MPX implementálása
...	...	**...**	...