1. Analízis modell kidolgozása
   1. Objektum katalógus

[Minden, a feladatban szereplő objektum rövid, egy-két bekezdés hosszú ismertetése. Meg kell jelenjen minden objektumhoz, hogy mi a felelőssége. Informális leírás, ezért nem kell foglalkozni az örökléssel, az interfészekkel, az absztrakt osztályokkal, a segédosztályokkal.]

* + 1. Objektum1

[Felelősség informális leírása]

* + 1. Objektum2

[Felelősség informális leírása]

* 1. 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.]

* + 1. DigitalBoard
       - Felelősség

Betölti az áramkört, nyilvántartja a DigitalObject-eket, felépíti a köztük lévő hierarchiát. Gerjeszti vagy megállítja a szimulációt.

* + - * Ősosztályok

nincs

* + - * Interfészek

nincs

* + - * Attribútumok
* **playing**:boolean Itt tartjuk számon, hogy fut-e a szimuláció. Értéke igaz,ha fut a szimuláció, ill. hamis ha nem fut.
* **speed**:int A generátor léptetési sebességét adja meg.
  + - * Metódusok
* **void LoadBoard(String file)**: betölti az áramkört egy fájlból. Felépíti a hierarchiát, mely majd a szimuláció során lesz fontos, az adatfolyam kezeléséhez.
* **void Pause()**: Megállítja a szimulációt. Ilyenkor a kívánt vezeték értéke leolvasható.
* **void Play()**: Elindítja a szimulációt, a **speed** változóban megadott értéknek megfelelően, adott ütemben lépteti a generátort/szekvenciát úgy, hogy Step()-et hív, ezzel szimulálja a hálózatot.
* **void setSpeed(int \_speed)**: Beállítja a **speed** változó értékét.
* **void Step()**: Szimulációs lépés, melynek hatására a generátor kiadja következő értékét és az összes DigitalObject elemre meghívja a Step() függvényt, a hierarchiának megfelelő sorrendben./\*A Step count részét,visszacsatolás stb. majd adott Step implementácónál mesélem el\*/
* **void Stop()**: Megállítja a szimulációt, az értékek/jelgenerátor állása alaphelyzetbe állítódik.

**//Kell még hierarchia tárolása**

**//Leszármazott osztályoknál csak implementált függvényeket írom be,ill. csak extra atrributumokat**

* + 1. Gate
       - Felelősség
       - Ősosztályok

Component->DigitalObject

* + - * Interfészek

nincs

* + - * Attribútumok

Az ősosztályok attribútumain kívül nincs más attribútuma.

* + - * Metódusok

Az ősosztályok metódusain kívül nincs más szolgáltatása.

* + 1. Inverter
       - Felelősség

A bemeneti jelet transzformálja az ellenkezőjére. Bináris esetben a hamisat igazba, igazat hamisba.

* + - * Ősosztályok

Component->DigitalObject->Gate

* + - * Interfészek

nincs

* + - * Attribútumok

Az ősosztályok attribútumain kívül nincs más attribútuma.

* + - * Metódusok

**void Count():** Az inverter kimenetét számolja ki a bemenete alapján. A **wireIn** változó **value** változóin elvégzi a logikai műveletét és **wireOut** változó értékét beállítja e szerint.

**void Step():**Az adott szimulációs ütemet szimulálja adott áramköri elemre. Ha a hálózat tartalmaz visszacsatolást és ez érinti az inverkaput úgy, hogy annak kimeneti értéke megváltozik, nem egyenlő a **sameAsPrevious** értékével, akkor újraszámolja a kimenetén lévő többi elem értékét **Count()** –al. Ha a visszacsatolás miatt ezt háromnál többször kell, hogy megismételje, akkor a hálózat instabil(hogy jelezzük?).

* + 1. orGate
       - Felelősség

Ha bármelyik bemeneti érétke igaz, akkor a kimeneti értékére is igazat rak ki. Ha minden bemeneti értéke hamis, akkor a kimenetére is hamisat ad ki.

* + - * Ősosztályok

Component->DigitalObject->Gate

* + - * Interfészek

nincs

* + - * Attribútumok

Az ősosztályok attribútumain kívül nincs más attribútuma.

* + - * Metódusok

**void Count():** A VAGY kapu kimenetét számolja ki a bemeneti alapján. A **wireIn** változó **value** változóin elvégzi a logikai műveletét és **wireOut** változó értékét beállítja e szerint.

**void Step():** Az adott szimulációs ütemet szimulálja adott áramköri elemre. Ha a hálózat tartalmaz visszacsatolást és ez érinti a VAGY kaput úgy, hogy annak kimeneti értéke megváltozik, nem egyenlő a **sameAsPrevious** értékével, akkor újraszámolja a kimenetén lévő többi elem értékét **Count()** –al. Ha a visszacsatolás miatt ezt háromnál többször kell, hogy megismételje, akkor a hálózat instabil(hogy jelezzük?).

* + 1. andGate
       - Felelősség

Ha mindegyik bemeneti érétke igaz, akkor a kimeneti értékére is igazat rak ki. Ha bármelyik bemeneti értéke hamis, akkor a kimenetére is hamisat ad ki.

* + - * Ősosztályok

Component->DigitalObject->Gate

* + - * Interfészek

nincs

* + - * Attribútumok

Az ősosztályok attribútumain kívül nincs más attribútuma.

* + - * Metódusok

**void Count():** Az ÉS kapu kimenetét számolja ki a bemeneti alapján. A **wireIn** változó **value** változóin elvégzi a logikai műveletét és **wireOut** változó értékét beállítja e szerint.

**void Step():** Az adott szimulációs ütemet szimulálja adott áramköri elemre. Ha a hálózat tartalmaz visszacsatolást és ez érinti az ÉS kaput úgy, hogy annak kimeneti értéke megváltozik, nem egyenlő a **sameAsPrevious** értékével, akkor újraszámolja a kimenetén lévő többi elem értékét **Count()** –al. Ha a visszacsatolás miatt ezt háromnál többször kell, hogy megismételje, akkor a hálózat instabil(hogy jelezzük?).

* + 1. Output
       - Felelősség
       - Ősosztályok

Component->DigitalObject

* + - * Interfészek

nincs

* + - * Attribútumok

Az ősosztályok attribútumain kívül nincs más attribútuma.

* + - * Metódusok

Az ősosztályok szolgáltatásain kívül nincs más metódusa.

* + 1. LED
       - Felelősség

Bemeneti értékét kijelzi a felhasználónak.

* + - * Ősosztályok

Component->DigitalObject->Output

* + - * Interfészek

nincs

* + - * Attribútumok

Az ősosztályok attribútumain kívül nincs más attribútuma.

* + - * Metódusok

**void Count():** Mivel a Led osztály nem változtat a bemenet értékén ez a függvény nem csinál semmit it.

**void Step():** Az adott szimulációs ütemet szimulálja adott áramköri elemre. A LED kijelzi az értékét (Grafikus megvalósításnál színt vált, más esetben szövegesen írja ki.)

* 1. Statikus struktúra diagramok

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

* 1. 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.]

* 1. State-chartok

[Csak azokhoz az osztályokhoz, ahol van értelme. Egyetlen állapotból álló state-chartok ne szerepeljenek. A játék működését bemutató state-chart-ot készíteni tilos.]

* 1. Napló

|  |  |  |  |
| --- | --- | --- | --- |
| **Kezdet** | **Időtartam** | **Résztvevők** | **Leírás** |
| 2010.03.21. 18:00 | 2,5 óra | Horváth  Németh  Tóth  Oláh | Értekezlet.  Döntés: Horváth elkészíti az osztálydiagramot, Oláh a use-case leírásokat. |
| 2010.03.23. 23:00 | 5 óra | Németh | Tevékenység: Németh implementálja a tesztelő programokat. |
| … | … | … | … |