Logikai modulok szimulálása konzolon

Ez a program logikai kapukból álló modulok szimulálására szolgál. Lehet benne létrehozni modulokat. Szimulálni inputok változását, és a program kiírja az inputok és outputok értékeit a szimulálás folyamán. Le lehet menteni egy fájlba, hogy milyen modulokat hozott létre a felhasználó, milyen inputokat adott meg, meddig futtatta a szimulációt, valamint milyen korábbi mentéseket olvasott be.

Utasítások=inputok:

1. modul létrehozása:

forma: <név>:<leírás>

1.1. egyszerű modul:
leírás: <érzékenység><hozzárendelés>,<érzékenység><hozzárendelés>,…
név megadása, minden outputra: inputoktól függésének megadása, ez a függés tartalmazhat érzékenységi listát: csak akkor változzon az output, ha az ebben a listában megadott inputok egyike változott, elhagyás esetén bármelyik a hozzárendelésben szereplő input változásakor lefut a hozzárendelés kiértékelése, és módosulhat az output.

1.2. komplex modul:
leírás: <modulnév>(<inputok>,<outputok>)<modulnév>(<inputok>,<outputok>)…
név megadása, használni kívánt modulok és bekötéseik

2. végrehajtás:

forma: <inputok><futtatásszám>< módosítók>
három részből áll melyek mindegyike opcionális, de sorrendjük kötött

2.1. inputok állítása: kis betű: az adott betűvel jelölt _main modulba vezető jel 0-ra (hamis) állítása, nagy betű: az adott betűvel jelölt _main modulba vezető jel 1-re (igaz) állítása. A be nem állított értékek bizonytalan állapotban, ’?’-vel jelölve jelennek meg.

2.2. futtatásszám megadása: egy pozitív integer. ha egy végrehajtás parancsnak van ilyen eleme, akkor ennyiszer fogja lefuttatni az éppen aktuális várakozó modulokat. Mindig, amikor egy modul outputjának értéke változik/ "inputok állítása" történik az adott outputra kötött modulok/_main modul felkerül a várakozó modulok listájára. Lényegében ekkor történik a szimuláció.

2.3. módosítók megadása:

2.3.1.’+’-jel: minden lefutásnál kiírja a _main modul portjainak értékeit,

2.3.2.’-’-jel: bármennyire nagy a lefutások száma, csak a végén írja ki a _main modul portjainak értékeit.

2.3.3.’%’-jel: konzol törlése, csak az átláthatóság érdekében, a háttérben semmit sem töröl, csak a konzolt tisztítja meg.

2.3.4.’!’-jel: kilép a programból.

3. fájlból beolvasás:

forma: <<fájlnév+kit>
beolvasandó fájl neve + kiterjesztése

A fájl ugyan olyan parancsokat tartalmazhat, mint amiket a konzolon adunk meg, a fájlba írás kivételével, amit nem tartalmazhat, valamint nem tartalmazhatja önmaga beolvasását, de más fájlokét igen (körkörös beolvasást sem). Ha nem sikerül megnyitni a fájlt kiírja, hogy nem sikerült.

4. fájlba írás:

forma: ><fájlnév+kit>
"ki" fájl neve + kiterjesztése.

Minden utasítás, amit kiadtunk/ fájlból hajtódott végre, (fájlba írások kivételével) egyesével megkérdezésre kerülnek, hogy bekerüljenek-e a "ki" fájlba 'I'(nagy i): igen, bármi más: nem.

Használat:

A fent felsorolt utasításokat lehet kiadni a konzolon, amik végrehajtódnak, vagy kiírja milyen hiba miatt nem hajthatók végre. Hibák kiírásánál használt nyelvezet: simp_modul: egyszerű modul, komp_modul: komplex modul

modul hibák:

1. nincs : jel

   Minden modul definiálásánál a név után kell egy ':'-jel, hogy honnan kezdődik a modul viselkedésének leírása

2. nem _ val kezdodo nev

   minden modul névnek '_'-vel kell kezdődnie

3. rossz karakter a nevben

   A név a következő karaktereket tartalmazhatja: „!#$%*+-./<=>?@_{}" valamint az angol abc ki és nagybetűit plusz a számokat

4. mar foglalt nev

   Egy nevet csak egy modulnak lehet adni

5. masodik kivezetes komp_modul-ban

   Minden kivezetés legyen az output vagy belő vezetékezés, csak egy helyről kaphatja az értékét

6. nem jo bekotes komp_modul-ban

   Az inputok az egymás után következő angol kisbetűivel vannak jelölve, valamint az outputok az első nem csak inputként használt kisbetűtől vannak sorban. Nem lehet egy modul inputjai 'a' és 'g' és ugyanakkor outputja 'b'; Ez helyesen 'ab' input és 'c' output.

   Ez a hiba jelentkezik akkor is, ha a nagy betűknél (belső vezetékezésben) van hiba: csak inputként van használva az egyik vezeték, csak outputként van használva (ha nem akarjuk használni egy modul outputját, azt '-' karakterrel jelezhetjük), nincs minden betű felhasználva a legnagyobb betű előtt (ABD van használva, de C nem).

7. nem letezo modul komp_modul-ban

   Nem definiált modult próbál használni egy komplex modulban

8. rossz inputszam komp_modul-ban

   Az egyik használni kívánt modulnak nem egyezik meg az input száma a '(' és a ',' közé írt karakterek számával

9. rossz outputszam komp_modul-ban

   Az egyik használni kívánt modulnak nem egyezik meg az output száma a ’,’ és a ’)’ közé írt karakterek számával

10. nem jo bekotes simp_modul -ban

    Az inputok az egymás után következő angol kisbetűivel vannak jelölve, ’abd’ használata rossz mert kimarad a ’c’

11. nem jo forma simp_modul -ban

    Az érzékenységi listában csak inputok(angol kisbetűk) állhatnak, az után viszont érték (input vagy konstans) és művelet csak felváltva állhat. Ez alól kivétel a ’~’ ami állhat érték elé és művelet elé is, de önmaga elé nem. Ezeknek zárójelezéstől függetlenül teljesülniük kell. pl.: ~(~a)) hibás, ab&c hibás, de ~a~&~b jó

12. rossz karakterek a simp_modul-ban

    az egyszerű modulban csak ’[]’-jelek (érzékenységi lista megadása), angol abc kisbetűi (inputok), ’()’ zárójelek(műveleti sorrend állítása), ’01’ (konstansok) és ’&|^~ ’ karakterek(és, vagy, kizáró vagy, nem), valamint ’,’-jel (outputok szétválasztása) használhatók.

További tudnivalók:

Az utasítással definiált modulok csak template-ek, pl.: egy komplex modul tartalmazhat egy adott modulból többet, nem kell hozzá többször definiálni egy-egy modult. Ez alól kivétel a _main modul, ami rögtön definiálás után példányosodik.

A program teljes használatára le van tiltva a space, nem szabad használni se moduloknál se sehol.

Fontos, hogy komplex modul csak olyan modulokat tartalmazhat, amik már definiálva lettek. A _main nevű modul lesz a szimuláció alapja, neki állíthatjuk az inputjait és neki látjuk az outputjait, így tehát végrehajtás utasítást a _main modul létrehozása előtt nincs nagyon értelme kiadni, de lehetséges.

* kiírások még változhatnak, de tartalmilag nem változtatok rajtuk, maximum szóhasználatilag.

Modul példák:

1. egyszerű modul érzékenység nélkül:

_and:a&b

Definiálja a _and nevű modult, hogy az első outputja: az első és második inputján kapott jelek és kapcsolatát adja ki.

_half_adder:a^b,a&b

Definiálja a _half_adder nevű modult, hogy az első outputja: az első és második inputján kapott értékek kizáró vagy kapcsolatát adja ki, és második outputja: az első és második inputján kapott értékek és kapcsolatát adja ki.

2. érzékenységgel:

_d_flipflop:[b]a

Definiálja a _d_flipflop nevű modult, hogy az első outputja: felvegye az első inputja értékét, amikor a második input megváltozik (fel és leszálló ágban is változásként van érzékelve).

3. komplex modul

tegyük fel, hogy a _and:a&b és a _xor:a^b utasításokat már kiadták ekkor a _half_adder modul így is kinézhet:

_half_adder:_and(ab,d)_xor(ab,c)

ekkor az _and modul első inputjára bekötjük a _half_adder első inputját(a) és második inputjára, a másodikat(b), valamint az első outputjára a _half_adder második outputját(d) az _xor modul inputjait ugyan úgy kötjük be, mint az _and modul esetében, viszont ennek az outputját a _half_adder első outputjára(c) kötjük

valamint a _full_adder, függetlenül attól, hogy hogy lett a _half_adder definiálva:

_full_adder:_half_adder(ab,AB)_half_adder(Ac,dC)_xor(BC,e)

Ekkor az első _half_adder modul első két inputja a _full_adder modul első két inputjára van kötve, outputjai pedig a belső vezetékek (nagy betűvel jelölt kívülről el nem érhető vezetékrendszer, a komplex modulokban) első kettőjére. A második _half_adder modul első inputja a belső vezetékezés első vezetékére van kötve, második bemenete pedig a _full_adder modul harmadik bemenetére. Első kivezetése a _full_adder modul első kivezetésére, második kimenete a belső vezetékezés harmadikjára van kötve. Végül a két carry-t egy or vagy xor kapuval össze kell rakni, erre szolgál a _xor modul a végén, ami a két _half_adder modul carry-ének kizáró vagy kapcsolatát kiadja a _full_adder modul második kivezetésére.

4. _main modul

ami egy két 4 bites számot ad össze:

_main:_full_adder(dh0,mA)_full_adder(cgA,lB)_full_adder(bfB,kC)_full_adder(aeC,ji)

összeadja az abcd és efgh inputok értékét és kiadja az eredményt az ijklm outputokon. Az első _full_adder modul harmadik inputján egy konstans 0 van bekötve, mert csak a két számot akarjuk összeadni és nincs kezdeti carry.

Példa végrehajtás utasításokra:

A fenti _main definiálása esetén

be: abcdefgh

minden inputot 0ra állít, de nem történik lefutás

be: 10-

tíz lefutást hajt végre, de csak a végeredményt írja ki

ki:

```
┌────────┬─────┐
|abcdefgh|ijklm|
|00000000|00000|
└────────┴─────┘
```

be: abCDeFGh10+

ki:

```
┌────────┬─────┐
|abcdefgh|ijklm|
|00110110|00000|
|00110110|00000|
|00110110|00101|
|00110110|00001|
|00110110|00001|
|00110110|01001|
|00110110|01001|
|00110110|01001|
|00110110|01001|
|00110110|01001|
└────────┴─────┘
```

be: ABEH10

ki:

```
┌────────┬─────┐
|abcdefgh|ijklm|
|11111111|01001|
|11111111|01001|
|11111111|11100|
|11111111|10110|
|11111111|11110|
|11111111|11110|
|11111111|11110|
|11111111|11110|
|11111111|11110|
|11111111|11110|
└────────┴─────┘
```

Példa mentésre

be: >save.txt

```
_and:a&b
I/N: I
_half_adder:a^b,a&b
I/N: I
_xor:a^b
I/N: I
_full_adder:_half_adder(ab,AB)_half_adder(Ac,dC)_xor(BC,e)
I/N: I
_main:_full_adder(dh0,mA)_full_adder(cgA,lB)_full_adder(bfB,kC)_full_adder(aeC,ji)
I/N: I
abcdefgh
I/N: N
10-
I/N: N
abCDeFGh10+
I/N: I
ABEH10
I/N: N
```

ekkor a save.txt tartalma:

```
_and:a&b
_half_adder:a^b,a&b
_xor:a^b
_full_adder:_half_adder(ab,AB)_half_adder(Ac,dC)_xor(BC,e)
_main:_full_adder(dh0,mA)_full_adder(cgA,lB)_full_adder(bfB,kC)_full_adder(aeC,ji)
abCDeFGh10+
```

Példa beolvasásra

amikor megnyitjuk a programot beolvashatjuk a "mentést"

be: <save.txt

```
sikeres megnyitas
main setted
┌────────┬─────┐
|abcdefgh|ijklm|
|00110110|?????|
|00110110|?????|
|00110110|????1|
|00110110|0???1|
|00110110|0?001|
|00110110|0?001|
|00110110|01001|
|00110110|01001|
|00110110|01001|
|00110110|01001|
└────────┴─────┘
```