

## Bedrótozva – Logikai áramkörök szimulációja Linux alatt

Azok a Linux felhasználók, akik nemcsak egyszerű hobbiként űzik a számítástechnikát, hanem intézményesített keretek között is van lehetőségük tanulni róla, előbb-utóbb találkozhatnak a logikai függvények és áramkörök, bővebben a digitális technika varázslatos világával. Kereskedelmi szimulációs eszközök sokasága lelhető fel, melyek jó része platformfüggetlen, de mitévő legyen az a lelkes halandó, aki csupán egyszerű logikai áramköröket szeretne szimulálni?

■ Ez a cikk bevallottan szűk réteghez szól: ha a nyájas olvasó a flip-flop szóról a jin-jangra, a regiszterről pedig határidőnaplóra asszociál, akkor érdemes lapozni: a többi cikk talán több izgalmat tartogat!

No de mi ez a mogorva kiszólás?

Csupán tudni kell, hogy a most bemutatandó programok használatában kizárólag a digitális technika elemi ismerete segítségével tudunk egyről a kettőre jutni.

Ha ebben a témakörben kicsit is elmélyedünk, óhatatlanul szükségünk lesz arra, hogy az elméletben, papíron szépen mutató függvényeket és kapcsolásokat kipróbáljuk. Két út kínálkozik: a helyi elektronikai boltból beszerezni a kívánt alkatrészeket és a szó szoros értelmében megépíteni a kapcsolást vagy egy szimulációs-tervező környezetben dolgozva néhány egérgattintás után megcsodálni a működő rendszert. Az első verzió is tartogat érdekességeket, de az már végképp a magazin profilja által felállított kereteket feszegetné, így most egeret és billentyűzetet ragadva lássunk munkához!

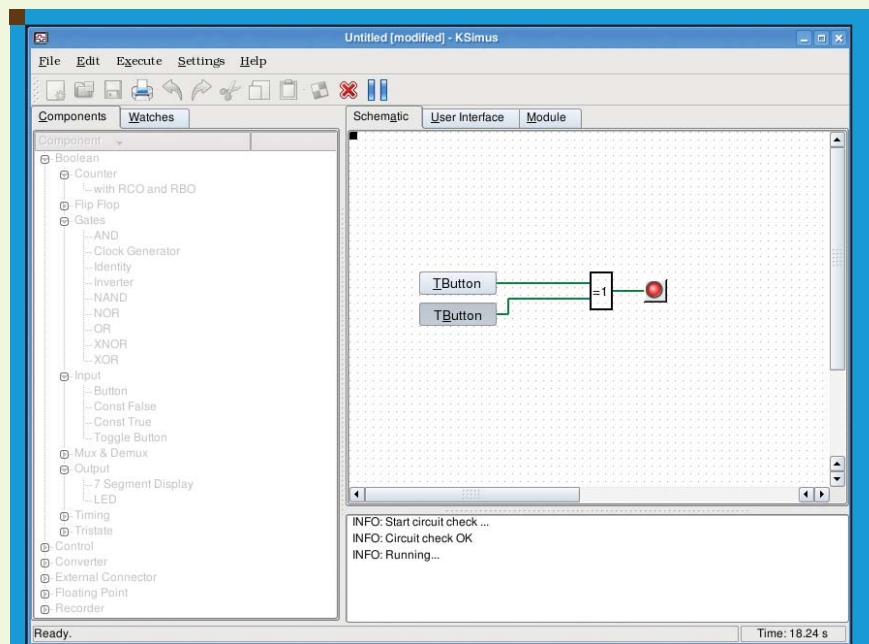
### Egy egyszerű megoldás

Elsőként a *KSImus* szoftverrel fogunk megismerkedni, amely *Rasmus Diekenbrock* munkája. Ezzel a szoftverrel pontosan azt kapjuk, amire a témával most ismerkedő felhasználónak szüksége lehet: könnyen és

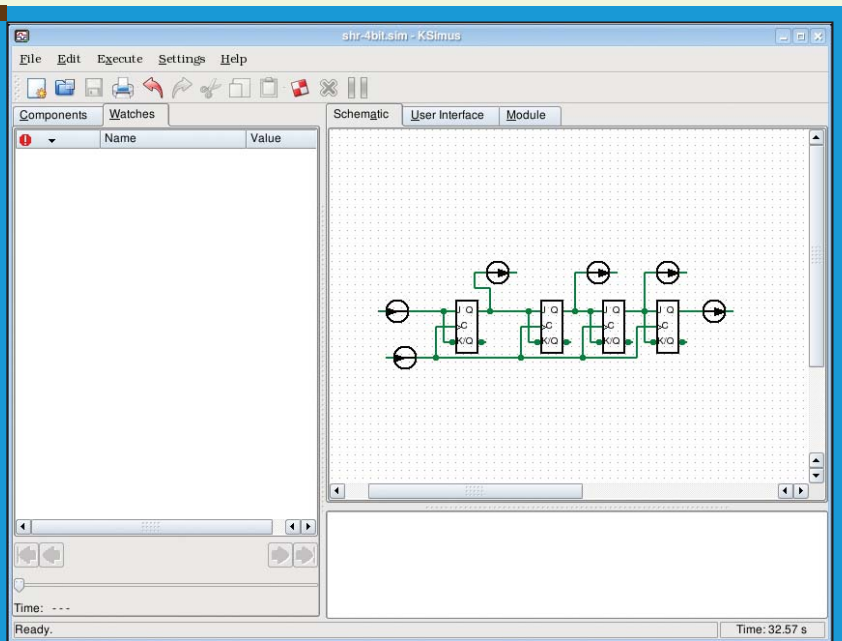
gyorsan össze tudunk rakni egy egyszerűbb kapcsolást, de akkor sem hagy cserben minket, amikor már összetettebb feladatok felé kacsingatunk. A 0.3.6-os verziószámot viselő csomagot, ha nem része a disztribúciónknak, a <http://ksimus.berlios.de/> címről tudjuk letölteni.

Kicsit jártasabb olvasóink biztos sejtik, hogy a K betűvel kezdődő név a *KDE* könyvtáraktól való függőséggel jár együtt és ez így is van. *Gtk* alapú logi-

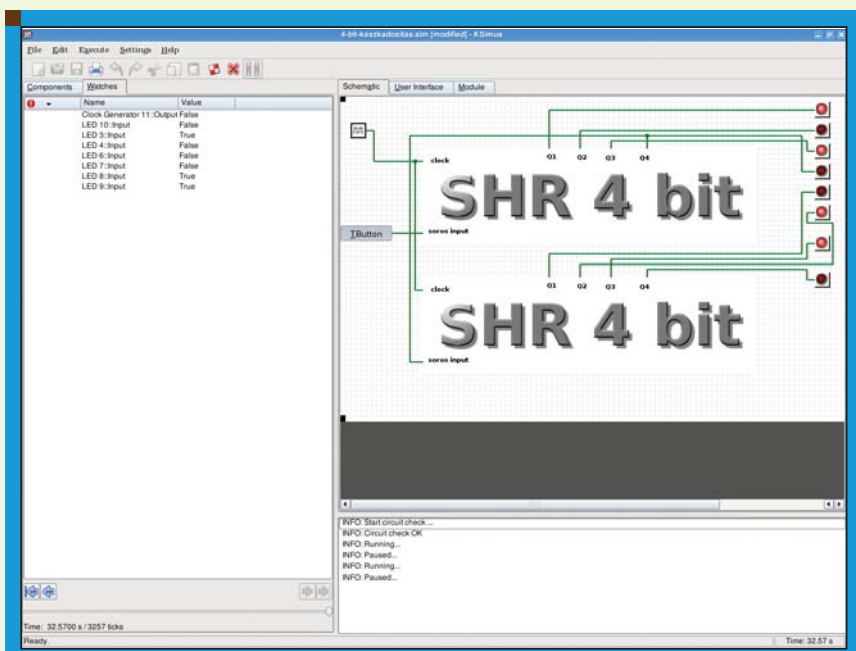
kai szimulátor létezik (*tlogsim*), de sokkal kevésbé funkciókban gazdag, mint *KDE*-s társai. A *KSImus* indulása után szembeötlő, hogy habár a rendszeren a magyar locale be volt állítva, angol nyelvű maradt a felület. Ezen persze bárki könnyedén segíthet, hiszen a szabad programok lefordításában akárki részt vehet. Még az is előfordulhat, hogy mire megjelenik a cikk, elkészül a fordítása a programnak.



■ 1. ábra Első összeállított egyszerű kapcsolásunk



■ 2. ábra Shift regiszter kapcsolási rajzon, felkészítve modulként való felhasználásra



■ 3. ábra A shift regiszter modul egy másik kapcsolásban felhasználva egy 8-bites shift regiszter céljára

A felhasználói felület nem bonyolult: a komponensek listája és a kapcsolási rajz tervezőfelülete tűnik elő indítás-kor. A komponensek listájában a *Boolean* csoport az, ami az első szárnypróbálgatásokhoz kell. A tankönyvek, jegyzetek és diák lapjairól oly jól ismert alkatrészek sorakoznak itt: különféle kapuk, flip-flopok, szám-lálók, multiplexerek és így tovább. A komponensek gyűjteménye egyáltalán nem olyan teljes, mint némely

kereskedelmi szoftveré, de ez két okból nem baj. Egyfelől az kezdeti lépések megtételéhez több mint elegendő a választék, másrészt ami hiányzik, elkészíthető. A program nagy erőssége, hogy új komponenseket (ezeket moduloknak nevezi a program), tetszőleges belső kapcsolással magunk is összeállíthatunk. Így hát nem baj, hogy shift regiszter mondjuk nincs a felsorolásban, kis erőfeszítéssel csinálhatunk egyet, amit

aztán akárhányszor felhasználhatunk. Most már tényleg lássunk munkához! A komponensek *Boolean/Gates*, *Boolean/Input* és *Boolean/Output* csoportjaiból van szükségünk elemekre egy elő példára. A komponens nevére történő kattintással az adott elemet beszúráshoz kijelöltük, utána ha a *Schematic* fülön lévő lapon kattintunk egyet, létrejön az elem most már ténylegesen a kapcsolási rajzunkon. Tegyük egy próbát az *XOR*-ral, a kizáró vagy logikai kapcsolat kapujával! Egy kapu önmagában mit sem ér, most a *Toggle Button*-ból helyezünk el két darabot, aztán pedig a *LED*-ből egyet. Néhány programban az alkatrészek összedrótozása precíz egérmozgatást igényel, itt a két ki illetve bemenetet (természetesen kimenetet bemenettel és viszont) kell kijelölnünk és a program elkészít egy megfelelő vezetékeztést. Nehezen bírható rá a program szerencsére, hogy átláthatatlan és kaotikus vonalakkal borítsa el a kapcsolást huzal címén, hacsak eleve az alkatrészeket logikátlanul helyeztük el. Ezután az *Execute/Start* menüpontot kiválasztva életre kel a kapcsolás, a gombok által adott logikai értékek függvényében a *LED* hol világít, hol nem. Most a példánkban csak akkor világít a *LED*, ha pontosan az egyik kapcsoló van benyomott állapotban.

**Modulok, ameddig a szem ellát**

Vegyünk egy shift regisztert! Azaz vennénk, de a komponensek között hiába keressük. Sebjaj, interneten shift regiszter kapcsolást lehet találni. Egy saját modult fogunk belőle készíteni. Tervezzünk tehát négy bites shift regisztert! Kell hozzá négy *D* flip-flop néhány ki és bemenet és összeköttetések. A [http://www.eelab.usyd.edu.au/digital\\_tutorial/part2/register02.html](http://www.eelab.usyd.edu.au/digital_tutorial/part2/register02.html) címen található kapcsolást készítjük el. Modult pontosan ugyanúgy kell csinálni, mint bármilyen más kapcsolási rajzot: elhelyezzük az alkatrészeket, ki- és bemeneteket, huzalozunk, tesztelünk. A program egyik hátránya máris szembeötlik: *D* flip-flop nincs, a *J-K* flip-flopot lehet átalakítani. Ezt úgy tehetjük meg, hogy a *J-K* flip-flop *K* lábára a *J*-re adott jel negáltját kötjük. Ha követjük a kapcsolási rajzot a megadott weblapon, úgy ezzel el is lehet készíteni a shift regisztert.

Hogy legyen belőle modul? A gombok és lámpák helyére tegyünk *External Connectorokat*, *Bool Inputot* a gombok, *Bool Outputot* a lámpák helyére. Ennyi az egész. Természetesen a belső, csak tesztcellán elhelyezett ledek helyére nem kell kivezetést tenni. Ezután a *Module* fölön eldönthetjük, hogyan is nézzen ki újonnan létrejött alkatrészünk. Akár saját képet is rakhatunk az alkatrész felületére. Ami ennél azért lényegesebb: itt rendelhetjük hozzá az egyes *Input/Output*-okat az alkatrész dobozán kivezetésekhez. Miután elmentettük a fájlt, máris felhasználható. Hozzunk létre egy új fájlt, aztán válasszuk az *Edit/Insert module*-t és szűrjük be a tervbe a saját alkatrészünket, akár több példányban. Én a példánál maradva méretnöveléssel 4-bites shift regiszterből készítettem egy 8-biteset. A példában szereplő megtervezett alkatrész nem teljes, egy tisztességes regisztert le kellene tudni törölni, amit itt nem lehet. A lehetőségek innen már csak a tervező fantáziájának szabta korlátokba ütközhetnek! Biztos ez? Akár *KDE* alá is találhatunk a fenténél nagyobb tudású rendszert, a *KTechLabot*

(☞ <http://ktechlab.org/>), amit szintén mindenkinek ajánlok kipróbálásra, mert logikai áramkörökön túl még nagyon sok másra is alkalmas. Mi az oka, hogy mégis a *KSimus* került terítékre? Egyszerűen az, hogy ezzel a szoftverrel gyorsan lehet látványos eredményeket elérni és nagyon gyorsan meg lehet tanulni a használatát. Persze akik nem csak játékszernek vagy tanulásuk eszközének tekintik a digitális tervezést, azok megmosolyogják ezeket a primitív eszközöket. Igazuk is van. Messzire menni mégsem kell, az *Icarus Verilog* majd minden disztribúcióban helyet kapott. Sőt mi több, *Linuxvilág* negyedik lapszámában *Michael Baxterrel*, az *Icarus Verilog* fordító készítőjével is olvashatunk egy érdekes interjút. Létezik, hogy nincs meg az összes lapszám kezdetektől fogva? Nagy hiba, de a ☞ [www.linuxvilag.hu](http://www.linuxvilag.hu) minden regisztrált felhasználónak elérhetővé tette a cikket (nemcsak ezt, hanem számtalan régebbi lapszám cikkét is). Akár kedvtelésből, akár szakmai alapokkal közeledünk a digitális tervezés felé, tapasztalhatjuk a szabad

szoftveres megoldások életképességét. Az egyszerűbb igényeknek a grafikus felületű, könnyen megtekinthető kapcsolási rajz alapú tervezők állnak, a tényleges tervezői munkához pedig a *Verilog* hardverleíró nyelv fordítója, az *Icarus* nyújt segítséget. A *Verilog*hoz egy rövid leírást találunk a ☞ <http://www.asic-world.com/verilog/veritut.html> webcímen. Míg a *KSimus* és a *KTechLab* digitális technikában járatosoknak néhány kattintás után kiismerhető, a *Verilog* olyan, mint egy programozási nyelv, hosszabb ideig tart, mire ténylegesen értelmes dologra tudjuk használni.



**Novák Áron**

(aaron@szentimre.hu)

BME-VIK-es hallgató, műkedvelő rendszergazda. Jelenleg leginkább a NetBeans-szel és mindenféle hordozható eszközzel foglalkozik, legálábbis mindazokkal, amelyeket meg lehet szólaltatni Linux alatt.

