Digitális Laboratóriumi Gyakorlatok

Jegyzőkönyv

2. gyakorlat

2024. február 29.

Stefán Kornél (TFRXIL) mérnökinformatika BSc
Vad Avar (I7NE8T) mérnökinformatika BSc

Elméleti összefoglaló

Az előző laboron megismerkedhettünk a logikai értékek fizikai manifesztációjával, ami amellett, hogy nagyon érdekes, nagyon hasznos is lesz számunkra. A félévben ezt az ismeretet fogjuk elmélyítenie, de az eheti gyakorlaton az alapokkal fogunk megismerkedni.

A logika két értéke nagyon szép és gyakorlatias, viszont magukban nem tudunk túl sok dolgot csinálni velük, ezért a későbbi összetett dolgok megvalósításához absztrakciót kell bevezetnünk, ez lesz a Boole algebra. Másnéven a logikai algebra azzal foglalkozik, hogy hogyan lehet igazság változókkal (olyan változókkal, amik tetszőleget logikai értéket vesznek fel) műveleteket elvégezni, hogy lehet ezen műveletekkel állításokat felírni, hogyan lehet egyes állításokat egyszerűsíteni.

Kezdjük is az alapokkal: az alap műveleteink, amiket chippek formájában is használni fogunk azok egy vagy két változóval vannak definiálva általában (de általánosíthatóak többre is). Az előző laboron a **nem** kapuval ismerkedtünk meg, vagy a logikai **negálás** művelettel, ami egy logikai értéket megfordít. Ennek a kiegészítő társa a kábel, ami a logikai értéket **ponálja**. Két változós szituációban már összetettebb a helyzet, mivel két változó két értéket vehet fel, így négy kimenet lehet és minden egyes kimenet egy logikai alapműveletnek számít. Általában ezeket a műveleteket egy táblázatban szoktuk ábrázolni, amiben az egyes sorok az egyes „logikai felállásokat” ábrázolják. Ilyenekkel találkozhatunk ebben a jegyzőkönyvben is majd. Az két legismertebb logikai kapunk a **vagy** és **és** kapuk, amelyek a logikai konjunktság vagy unió műveltet valósítják meg. A konjunkció művelet akkor eredményes igazat, ha mindkét paramétere igaz, emellett az unió akkor igaz, ha bármely bemenete igaz. Ezen kapuk egy változata az, ahol a kimeneti jel invertálva van **nem vagy** és **nem és**. Ezen túl logikai áramköröknél és informatikában elterjedt művelet még a **kizáró vagy**, ami akkor igaz, ha bármely bemenet igaz, de nem mind a kettő egyszerre.

A logika ábrázolására sok eszköz áll rendelkezésünkre. Az egyik módja a diszkrét matekon is tanult jel alapú, „matekos”, felírás, ahol a bementeink logikai változók lesznek és algebrai egyenletet írunk fel a kimenetekre. Ez nagyon kényelmes olyan szempontból, hogy az előbb említett igazságtáblás felírásnál az elemi műveletek elvégzésével tudjuk „darabonként” megoldani az egyenletet. Továbbá ez algoritmizálható, így nem is kell feltétlen kézzel csinálni. Ennél talán még sokkal fontosabb dolog az is, hogy matematikai azonosságok létezése miatt átalakíthatóak ezek az egyenletek, és el tudjuk azt érni, hogy kevesebb műveletet vegyenek igénybe, ami számolásnál (főleg, ha a gép végzi) nem tesz nagy különbséget, viszont, ha ez nekünk fizikailag több alkatrészt (és labor esetén kábelt) jelent, az nagyon is hasznos. Továbbá nyilván egy több milliós gyártósoron nem mindegy, hogy 2 vagy 4 darabot rakunk ugyan abból az alkatrészből.

A logikai formulák egy megvalósításhoz közelebbi, ámbár még nem valós, ábrázolási módja is létezik, ahol a bementeket és kimeneteket gombócokkal jelöljük és ezeket logikai kapukon keresztül összekötjük egymással. Ez a logikai rajz nem tartalmaz semmit a kapuk fizikai szükségeiről, csak a logikai bemenetüket és a logikai kimeneteiket. Emiatt könnyen átlátható és tervezhető, viszont a fizikai megvalósításhoz még ezt át kell alakítani egy elektromos áramkör tervvé, ami már nem egy megterhelő feladat, mivel metodikus művelet, viszont erősen idő igényes lehet. Ezen rajzoknak két féléjét ismerjük, az amerikai változatot, ahol különböző űrhajók formája határozza meg a logikai műveletet és az európai szabvány, ahol a négyzetek tartalma utal az elvégzett művelet kilétére. A jegyzőkönyvekben mind a kettő jelölés megtalálható lesz.

Feladatok

1. Feladat

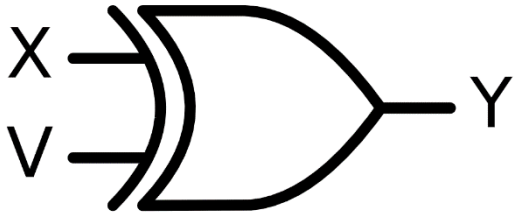
Valósítson meg egy programozható invertert, vagyis egy olyan kapcsolást, melyben egy **V** vezérlővonal segítségével adhatjuk meg, hogy invertáltja-e a bemenetre kapcsolt **X** jelet, vagy nem.

A hálózat bemenetei: **X, V**

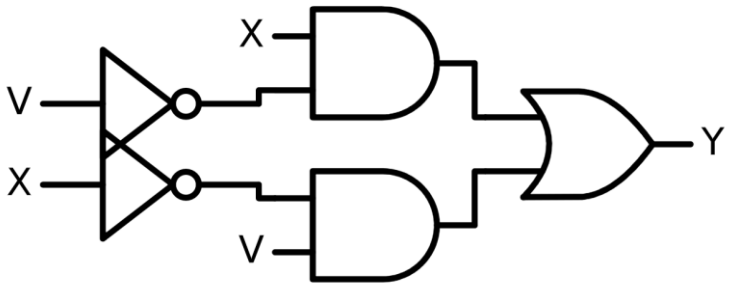
A hálózat kimenete: **Y**

X	V	Y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Logikai ábra



Megvalósított hálózat



2. Feladat

Valósítson meg egy multiplexert, mellyel egy **S** logikai jel segítségével a két, **X0** és **X1** bemenet közül választhatjuk ki, hogy melyik jelenjen meg a kimeneten!

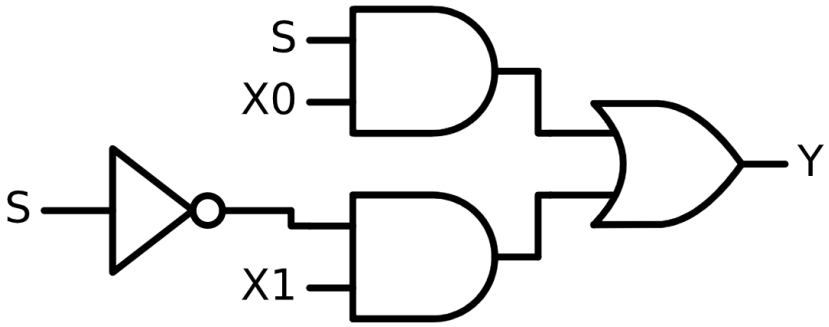
A hálózat bemenetei: **X0**, **X1**, **S**

A hálózat kimenete: **Y**

S	X0	X1	Y
0	0	X	**0**
0	1	X	**1**
1	X	0	**0**
1	X	1	**1**

Az igazságtáblában a bemeneteknél az X azt jelöli, hogy oda bármilyen érték kerülhet (0 vagy 1), a hálózat kimenete attól nem fog függeni.

Megvalósított kapcsolás



3. Feladat

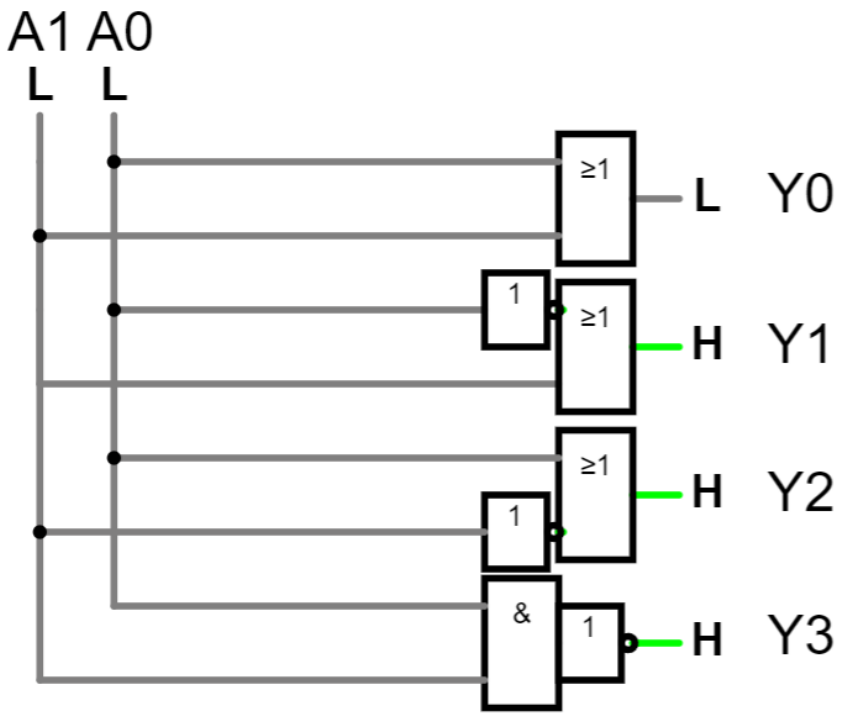
Valósítson meg egy 4-ből 1 dekódolót. A dekódoló úgy működik, hogy egy bináris szám értékének megfelelő sorszámú kimeneti jel lesz aktív. Mivel a 4 különböző érték két bittel állítható elő, ezért a bináris szám $\mathbf{A1} \cdot 2^1 + \mathbf{A0} \cdot 2^0$ alakú, tehát két bemenetre van szükség a négy $\mathbf{Y0}, \mathbf{Y1}, \mathbf{Y2}, \mathbf{Y3}$ kimeneti jel előállításához. A dekódoló kimenetei negatív logikát követsenek, vagyis a 0 jelentse az aktív állapotot, az 1 pedig az inaktívat (így mindig csak egy kimenet 0, a többi meg 1).

A hálózat bemenetei: $\mathbf{A0}, \mathbf{A1}$

A hálózat kimenete: $\mathbf{Y0}, \mathbf{Y1}, \mathbf{Y2}, \mathbf{Y3}$

A0	A1	Y0	Y1	Y2	Y3
0	0	**0**	**1**	**1**	**1**
0	1	**1**	**0**	**1**	**1**
1	0	**1**	**1**	**0**	**1**
1	1	**1**	**1**	**1**	**0**

Megvalósított kapcsolás/Szimulátor



4. Feladat

Valósítson meg egy négy bemenetű prioritásdekódolót! Az áramkör feladata, hogy megadja, hogy az **X0**, **X1**, **X2**, **X3** bemenetei közül melyik a legnagyobb prioritású (sorszámú), amelyik aktív, és az eredményt egy bináris számként szolgáltatja $\mathbf{Y1} \cdot 2^1 + \mathbf{Y0} \cdot 2^0$ formában. (A kisebb sorszámú bemenetek állapotát figyelmen kívül hagyja.). Ha például **X2** értéke 1, **X3** értéke 0 (**X0** és **X1** értéke ekkor nem számít, mert indexük kisebb, mint 2), akkor az index értéke 2, azaz a kimeneten $\mathbf{Y1} = 1$ és $\mathbf{Y0} = 0$ jelenik meg.

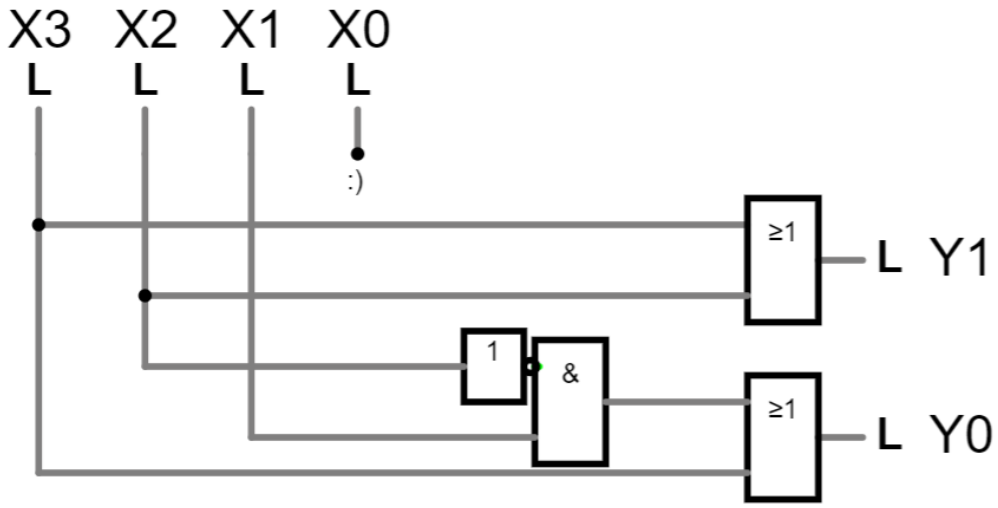
A hálózat bemenetei: **X0**, **X1**, **X2**, **X3**

A hálózat kimenete: **Y0**, **Y1**

X0	X1	X2	X3	Y0	Y1
X	0	0	0	**0**	**0**
X	1	0	0	**1**	**0**
X	X	1	0	**0**	**1**
X	X	X	1	**1**	**1**

Az igazságtáblában a bemeneteknél az X azt jelöli, hogy oda bármilyen érték kerülhet (0 vagy 1), a hálózat kimenete attól nem fog függeni.

Megvalósított kapcsolás/Szimulátor



Magyarázat

A maximum prioritásnál nem kell kezelni a többi esetet, mivel "úgy is minden fel van kapcsolva". A minimum prioritásnál nem kell semmit bekötni, és így "semmi nem fog bezavarni". Az **X2** bemenetet tudja zavarni az **X1** bemenet, ezért egy ellenőrző áramkört rakunk **X1** kimenetére (**Y0**), hogy ha az **X2** bemenet is be van kapcsolva, akkor kapcsoljon ki az **X1** kimenete (**Y0**).