

SALMON I PRO ŠKOLY

NÁVOD - ČÁST DRUHÁ
POČÍTAČOVÁ LOGIKA
A BINÁRNÍ ČÍSLA

Úvod

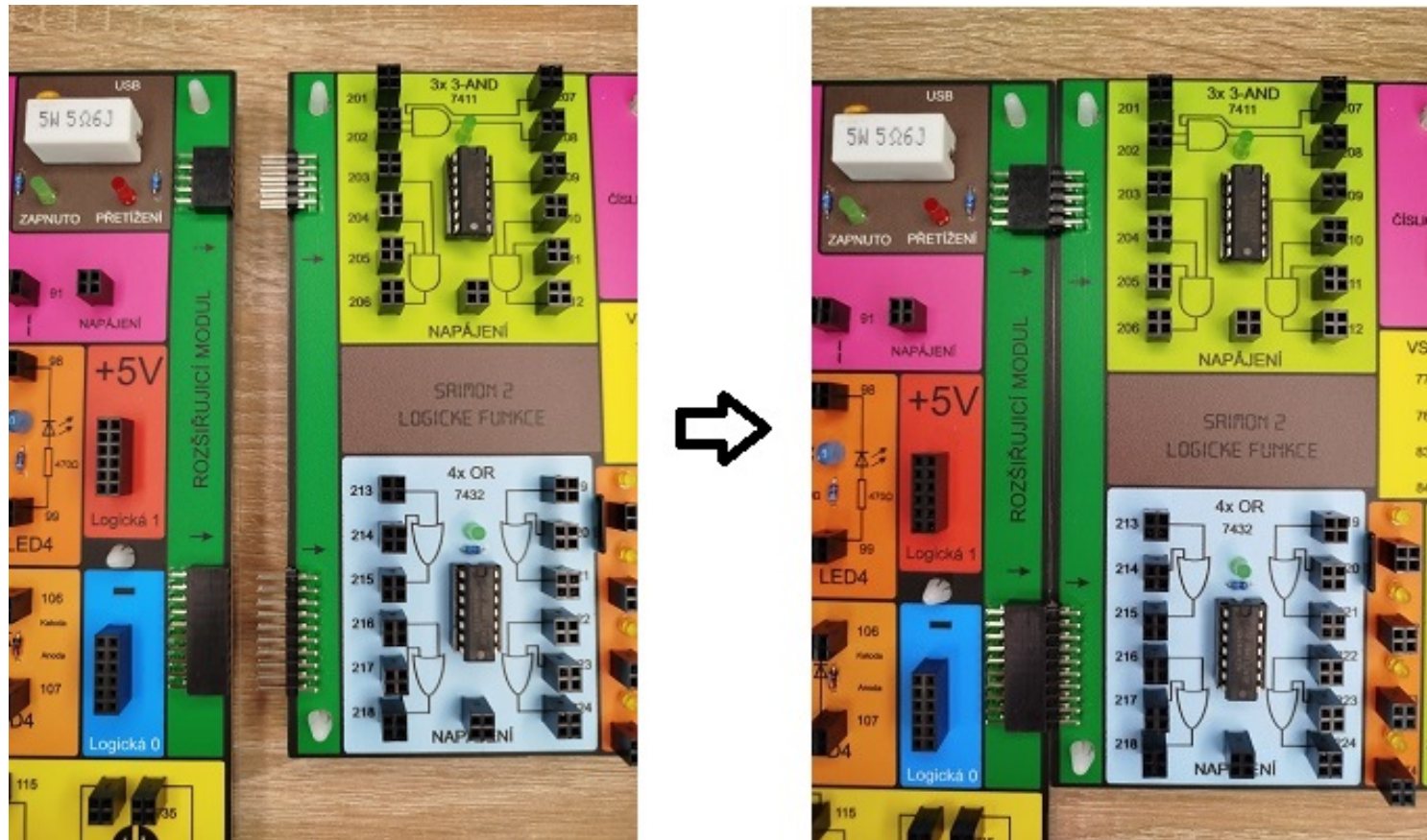
V této části návodu se budeme věnovat úvodu do konstrukce počítačů. Jak si počítač „pamatuje“ čísla. Co je to vlastně číslo. Naučíme se dvojkovou, neboli **binární soustavu** čísel. Ukážeme si, že existuje i šestnáctková, hexadecimální soustava a zobrazíme znaky na číslicovce.

Ukážeme si, jak funguje paměť. Máme k dispozici čtyř bitovou paměť v obvodu 74175. Sestavíme si i vlastní, pomocí hradla NAND 7400 a tranzistorů.

Nakonec si ukážeme, jak zapojit obvod časovače 555. Co takový integrovaný obvod umí, a jak to využijeme v našich obvodech. Jak zapojit obvody pro získání kvalitních hodinových pulsů.

Připojení Saimon 2

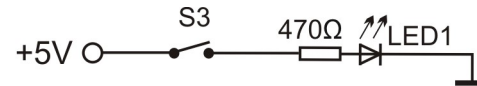
V případě, že vlastníte modul Saimon 2, ukážeme si, jak ho připojit k modulu Saimon 1. Saimon 2 se připojuje zasunutím vidlic konektorů na levé straně modulu do zdírek konektorů na pravé straně modulu Saimon 1. Viz obrázek.



Obrázek 0.1: Připojení modulu Saimon 2

Kapitola 1 - Jak počítač „vidí“ čísla

Začneme tím, jak počítač „vidí“, co je to vlastně číslo. Vrátime se na chvíli k jednomu z prvních obvodů z minulého dílu, ledka se spínačem.



Obrázek 1.1: Ledka se spínačem

Víme, že ledka svítí, když sepneme spínač. A když ho nesepneme, tak nesvítí. Já odteď budu říkat, že zmáčknuté tlačítko znamená jedničku. Nezmáčknuté znamená nulu. Stejně tak to, že ledka svítí, znamená 1. A když ledka nesvítí, znamená to 0. Tohle teď napíšu do tabulky

| S3 | LED1 | |
|----|------|---|
| 0 | 0 | Tlačítko nezmáčknuté = 0, LED nesvítí = 0 |
| 1 | 1 | Tlačítko zmáčknuté = 1, LED svítí = 1 |

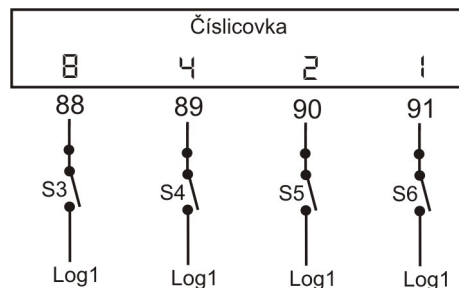
Obrázek 1.2: Tabulka hodnot obvodu se spínačem a ledkou

Když ledka svítí, tak skrz stisknuté tlačítko teče elektrický proud. Připomínám, že teče od plus, skrz rezistor, k vývodu ledky zvaný anoda, proteče ledkou a z vývodu katoda pak teče na mínus. Jinými slovy, když na anodu přivedeme plus, katoda je spojená s mínus, tak ledka svítí. A to znamená hodnotu 1.

V tomhle díle už pro nás plus bude nejen plus, ale také budeme říkat **Logická 1**, zkráceně **log 1**. Kdybychom na zdířku 93 přivedli mínus, tak LED1 svítit nebude, a tomu budeme říkat stav s hodnotou 0. Mínusu, nebo zemi, budeme říkat **Logická 0**, zkráceně **log 0**.

Teď to zopakuji s novým názvoslovím - když přivedeme hodnotu **log 1** na zdířku 93 u **LED1**, zdířka 92 bude vést na hodnotu **log 0**, tak bude ledka ve stavu 1 (bude svítit).

Jak počítač „vidí“ čísla. Počítače neumí pracovat s ničím jiným, než právě se stavy zapnuto a vypnuto, tedy že napětí je, nebo není. Říkáme $\log 0$ (není napětí) a $\log 1$ (je napětí). A pomocí toho umí dát dohromady třeba čísla. Zapojte následující schéma.



Obrázek 1.3: Zobrazení čísel na číslicovce

Úloha 1: Zapojení 123 - 126, 126 - 129, 129 - 132, 123 - $\log 1$, $\log 1$ - NAPÁJENÍ číslicovky, 122 - 88, 125 - 89, 128 - 90, 131 - 91

Stiskem spínače se nastaví hodnota $\log 1$ na vývody číslicovky. Když zmáčknete S4, které vede do zdířky 89, zobrazí se na číslicovce číslo 4. Digitální čtyřka je přímo pod zdířkou nakreslená. Stiskněte jiný spínač. Zobrazí se opět číslo pod zdířkou. Dále zkuste kombinace stisknutých tlačítek.

Výzva 1: Dokážete napsat číslo pro každou z kombinací uvedené v tabulce na obrázku 1.4? Která čísla v tabulce chybí a jaká by to byla kombinace nul a jedniček?

| S3 | S4 | S5 | S6 | ČÍSLO |
|----|----|----|----|-------|
| 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | |

Obrázek 1.4: Tabulka kombinací

Velmi rychle zjistíte, že kombinací, jak stisknout tlačítka, je víc, než je čísel. Čísel je 0,1,2,3,4,5,6,7,8,9. To je deset číslic (0 je první, 1 druhá, 9 desátá atd.). Kombinací celkem můžete najít 16. Aby se kombinace daly využít, tak se ty, pro které nám nezbyla číslice, zobrazují jako písmeno. A to

podle tabulky na obrázku 1.5 (Poznámka - písmena A,b,C,d,E,F se zobrazují pouze na HEX verzi školní stavebnice. U verze BCD se zobrazí prázdný znak).

| S3 | S4 | S5 | S6 | ČÍSLO |
|----|----|----|----|-------|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |

| S3 | S4 | S5 | S6 | ČÍSLO |
|----|----|----|----|-------|
| 1 | 0 | 0 | 0 | 8 |
| 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | A |
| 1 | 0 | 1 | 1 | b |
| 1 | 1 | 0 | 0 | C |
| 1 | 1 | 0 | 1 | d |
| 1 | 1 | 1 | 0 | E |
| 1 | 1 | 1 | 1 | F |

Obrázek 1.5: Tabulka zobrazení čísel a hexa-znaků

Místo čísla 10, na které bychom potřebovali už dva číselné znaky (1 a 0), zobrazíme písmeno A, aby nám stačil jeden znak. Místo 11 zobrazujeme písmenko b, 12 = C, 13 = d, 14 = E a 15 = F. Způsobu zobrazení čísel pomocí nul a jedniček se říká **dvojková**, nebo **binární soustava** čísel. Tu máme vymyšlenou právě proto, že počítače znají jen vypnuto = 0, zapnuto = 1. Když přidáme písmena ABCDEF k číslicím, abychom využili všechny kombinace nul a jedniček, tak tomu říkáme **šestnáctková**, nebo **hexadecimální soustava** čísel. Ta soustava čísel, ve které jsme zvyklí počítat my lidé, se jmenuje **desítková**, neboli **decimální** (někde se uvádí dekadická). Důvod je ten, že máme na ruku deset prstů a bylo jednoduché každému přiřadit číslici (kdybychom měli prstů šestnáct, tak by se nám s počítači dnes počítalo lépe).

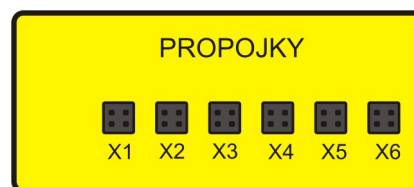
Binární soustava čísel bude pro nás nejdůležitější. Číslům 1010, 100, 1110, 111 atd. budeme říkat binární zápis čísla, a já budu v textu používat název binární číslo. Budu tím myslet takovýto zápis nul a jedniček. Těm jednotlivým jedničkám a nulám budeme říkat **bity**. Bit (z anglického výrazu BInary digiT) je pro nás právě tahle nejmenší jednotka, která může být 0 nebo 1. Číslo 1010 jsou 4 bity, 100 jsou 3 bity, 11 jsou dva, 10110 je 5 bitů a tak dále.

Zapojení se Saimon 2 – následující obvod můžeme zapojit i s modulem Saimon 2. Na něm je umístěná číslicovka zelené barvy.

Úloha 1a: Zapojení 122 - 125, 125 - 128, 128 - 131, 122 – log1

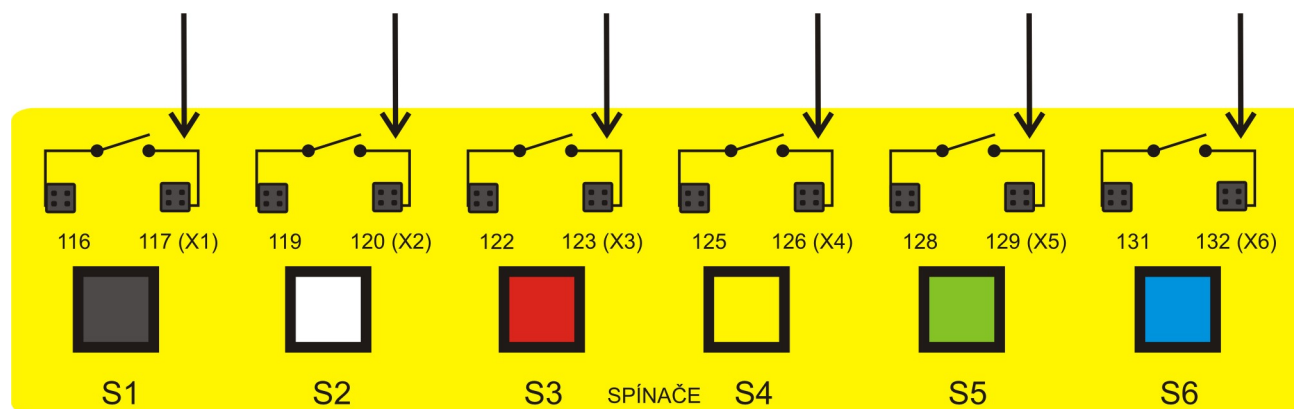
log1 - NAPÁJENÍ číslicovky (Saimon2), 123 - 259, 126 - 260, 129 - 261, 132 - 262

Když začnete zapojovat drátky od spínačů ke zdírkám číslicovky (259, 260, 261, 262), zjistíte, že potřebujete hodně dlouhé drátky. Abychom si to trochu ulehčili, použijeme PROPOJKY. Na modulu Saimon2 je pole zdírek označených X1, X2, X3, X4, X5, X6.



Obrázek 1.6: Propojky na modulu Saimon 2

Na hlavním modulu si všimněte, že zdíčky 117,120,123,126,129 a 132 mají ještě označení X1,X2,X3,X4,X5,X6.



Obrázek 1.7: Propojky na modulu Saimon 1

Jak propojky fungují? Ukážeme si na příkladu. Pokud chceme propojit například zdířky 123 a 259, potřebujeme buď dlouhý drátek, nebo využijeme propojky. Zdířka 123 má zároveň označení X3 v závorce. To znamená, že je vyvedena na propojku X3 na modulu Saimon 2. Stačí tedy zapojit drátek mezi zdířkami X3 - 259 na modulu Saimon 2. Zdířky stejného označení jsou mezi moduly **propojené** skrze konektory na boku modulů, pomocí kterých jste moduly spojili (je to vidět zespodu stavebnice). Propojky nám budou sloužit jako zkratky propojení mezi moduly, kde bychom jinak potřebovali dlouhé drátky.

Stejným způsobem fungují zdířky modrá Log0 a červená Log1 na modulu Saimon 2. Jsou to zdířky, které mají stejnou funkci, jako +5V a mínus na základním modulu Saimon 1. Pokud budeme chtít zapojit napájení integrovaných obvodů modulu Saimon 2, lze zapojit drátky do Log1 na Saimon2 – není třeba dlouhý drátek až do Saimon1.

Zapojení bude vypadat takto:

Úloha 1a: Zapojení 122 - 125, 125 - 128, 128 - 131, 122 – log1
log1(Saimon2) - NAPÁJENÍ číslicovky (Saimon2), X3 - 259, X4 - 260, X5 - 261, X6 - 262

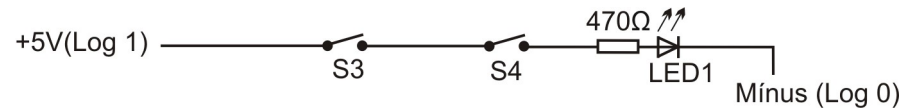
Jak to vypadá zapojené, se můžete podívat na obrázku 1.8.

Poznámka - všimněte si, že čísla zdířek na Saimon2 začínají vždy číslem větším, než 200. Je to z důvodu, abychom poznali podle čísla, jestli zdířku hledat na Saimon1, nebo Saimon2.

Kapitola 2 - Logické funkce

V této části návodu Saimon 1 pro školy si uvedeme zjednodušený přehled logických funkcí, které budeme sestavovat pomocí logické funkce NAND. Detailní popis logických funkcí najdete v druhém díle návodu pro plnohodnotný modul Saimon 1, kde jsou popsány logické funkce modulů Saimon 1 a Saimon 2.

První logická funkce, se kterou se seznámíme, bude funkce **AND**. V překladu AND znamená "A zároveň". Představte si obvod, ve kterém budeme mít dva spínače a ledku. Dokážete vymyslet zapojení tak, aby se ledka rozsvítila jen v případě, že stisknete oba dva spínače najednou? Řešení je na obrázku 2.1.



Obrázek 2.1 - Funkce AND zapojená pomocí spínačů

Úloha 2.0: Zapojení 122 - log1, 123 - 125, 126 - 93, 92 - log0.

Je jasné vidět, že musíme stisknout **oba** spínače **zároveň**, aby obvodem mohl téci elektrický proud. Zapišeme všechny kombinace stisku spínačů do tabulky. Kdy bude LED1 svítit?

| S3 | S4 | LED1 |
|----|----|------|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



Obrázek 2.2: Tabulka funkce AND a její schematická značka

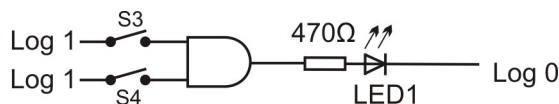
Z tabulky je vidět, že LED1 bude svítit jen v případě, že budou obě tlačítka ve stavu 1. To, co jsme právě vytvořili, se nazývá **logická funkce AND** neboli **logický součin**. Spínače S3 a S4 jsou pro nás **vstupy** logické funkce, LED je pro nás **výstup** logické funkce.

Ted' se na chvíli vrátím k integrovaným obvodům, které byly popsány v kapitole 9 prvního dílu návodu. Tam jsem mimo jiné psal, že integrovaný obvod má mnoho funkcí, podle jeho typu, a že uvnitř něj je spousta součástek. Nás bude nyní zajímat integrovaný obvod 7400. Obvod má popisek 4xNAND. Znamená to, že uvnitř jsou čtyři obvody logických funkcí NAND.

Logická funkce je pro naše účely **obvod**, který má jeden nebo více **vstupů** a jeden **výstup**. **Vstup logické funkce** je takové místo v obvodu, kam přivádíme hodnotu **log1** nebo **log0**. Obvod **vyhodnotí** vstupy a nastaví výstup na odpovídající hodnotu log0 nebo log1. Nastavení výstupu vždy provádí podle tabulky dané funkce. Například námi vytvořená funkce AND nastaví hodnotu výstupu podle hodnot vstupů, jak říká tabulka na obrázku 2.2.

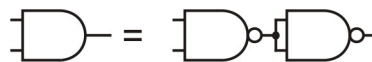
Vstupů může mít funkce i více. My zde máme dva a já je budu značit A a B. V jiné literatuře můžete najít značení například X1, X2. **Výstup logické funkce** je vždy jen **jeden**. Já ho budu značit například písmenem C (v jiné literatuře můžete najít jiné značení, například písmenkem Y).

Pokud tedy obvod nastavuje hodnotu výstupu podle hodnot vstupů tak, jak je uvedeno v tabulce na obrázku 2.2, nazýváme ho logická funkce AND a značíme ho schematickou značkou, která je na obrázku 2.2 nakreslená vpravo vedle tabulky. Funkci AND jsme zapojili pomocí spínačů a teď ji zapojíme pomocí integrovaného obvodu. Zapojení by vypadalo například jako na obrázku 2.3



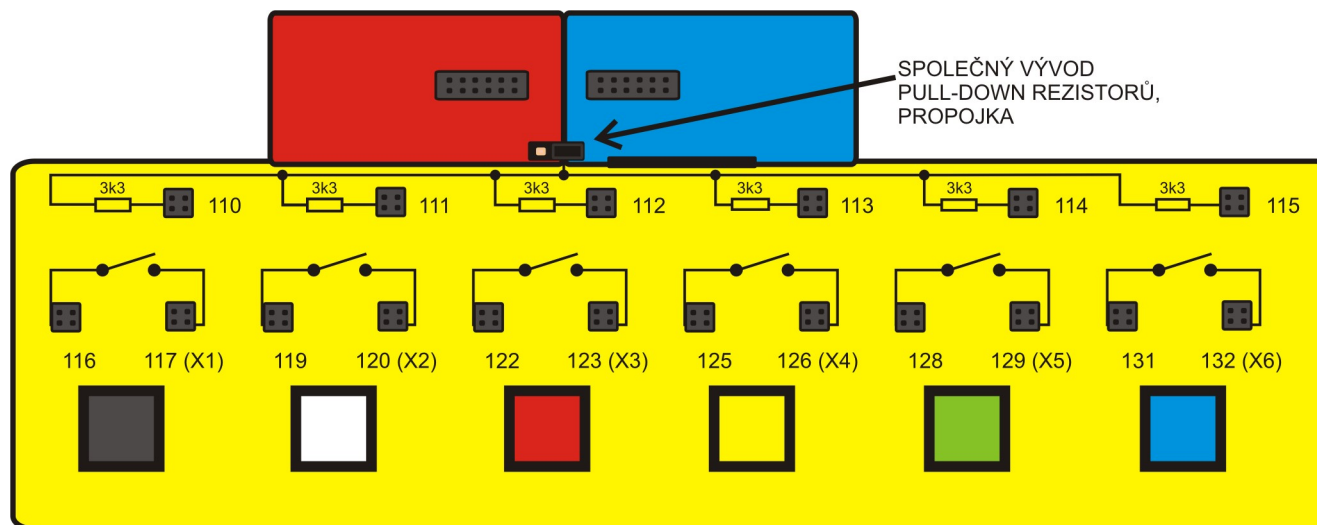
Obrázek 2.3: Funkce AND

Značku funkce AND ale na modulu Saimon 1 nenajdete (nachází se na modulu Saimon 2 v integrovaném obvodu 7408). Abychom mohli obvod zapojit, musíme si pomoci funkcí NAND. Jak se sestaví funkce AND ze dvou funkcí NAND je nakreslené na obrázku 2.4. Funkci NAND a princip sestavení funkce AND vysvětlím později, nyní to jen zapojíme.



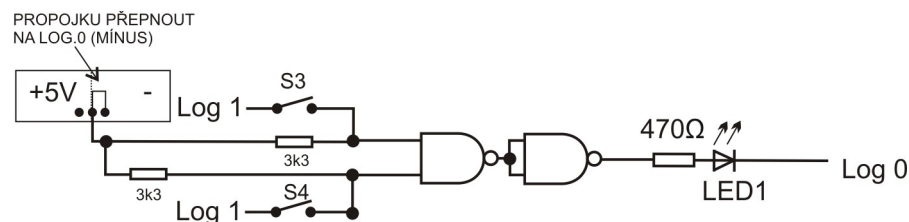
Obrázek 2.4: Funkce AND složená pomocí 2x NAND

Stisknutý spínač přivede na vstup log1. Když ale spínač nestiskneme, není na vstupu integrovaného obvodu nic. Obvod by vám fungoval špatně. Je třeba, aby na vstupech integrovaného obvodu byla vždy buď log0, nebo log1. To zajistíme pomocí rezistorů vyvedených na zdičky 110 až 115.



Obrázek 2.5 – Rezistory pull-down (pull-up)

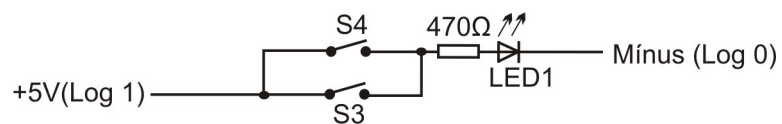
Na stavebnici je síť rezistorů s hodnotou 3k3 a jejich jeden konec je spojený. Ten je potřeba připojit na log0. Druhý konec rezistoru je třeba přivést na vstup logické funkce. Na obrázku 2.5 je vidět propojka a vývody rezistorů na zdičkách 110 až 115. Umístěním propojky (nachází mezi červeným a modrým polem plus a minus) vpravo zajistíme, aby při nestisknutém spínači rezistor přiváděl na vstup log0. Takovému rezistoru se říká pull-down rezistor (pokud bychom spojený konec rezistorů přivedli na log1, říká se mu pull-up rezistor). Výstup logické funkce můžeme sledovat například ledkou (svítí – výstup má hodnotu log1, nesvítí – výstup má hodnotu log0). Schéma obvodu s pull-down rezistory je na obrázku 2.6.



Obrázek 2.6: Zapojení vstupů a výstupů funkce AND pomocí NAND

Úloha 2.1: Zapojení 122 - log1, 122 - 125, 123 - 72, 126 - 71, 73 - 68, 68 - 69, 70 - 93, 92 - minus (log0), 123 - 112, 126 - 113, napájení 7400 - plus (log1), propojku rezistorů zapojit na minus.

Další funkce, se kterou se seznámíme, bude funkce **OR**. To v překladu znamená „nebo“. Zkuste opět vymyslet obvod, ve kterém budou dva spínače a ledka. Ta bude svítit, když bude alespoň jeden ze dvou spínačů sepnutý, nebo budou sepnuté oba dva. Řešení je na obrázku 2.7.



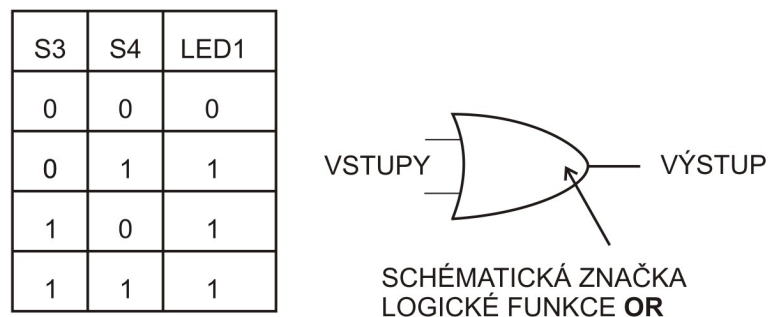
Obrázek 2.7: Funkce OR se spínači

Úloha 2.2: Zapojení 122 - log1, 122 - 125, 123 - 126, 126 - 93, 92 - log0

Je vidět, že když stiskneme jakékoliv tlačítko, tak proud obvodem poteče. Můžeme zmáčknout oba spínače, **nebo** jen jeden z nich. Takto vypadá **logická funkce OR** neboli **logický součet**. Zapojení spínačů se říká **paralelní** (jsou vedle sebe).

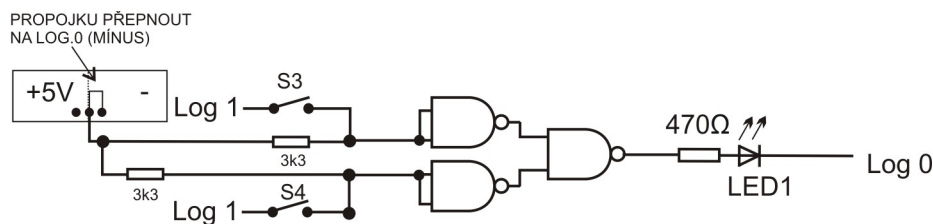
Výzva 2: Dokážete vymyslet obvod pro logickou funkci OR s více, než dvěma vstupy?

Tabulka funkce OR je na obrázku 2.8. Těmto tabulkám se říká **pravdivostní tabulky**. Log 1 pro nás také bude znamenat **pravda** (angl. **TRUE**) a log0 nebude lež, ale **nepravda** (angl. **FALSE**). Tabulky říkají, při jakých hodnotách vstupů jsou výstupy pravda nebo nepravda.



Obrázek 2.8: Pravdivostní tabulka funkce OR a její schematická značka

Funkci OR opět musíme sestavit z funkce NAND (funkce OR se nachází na modulu Saimon 2 v integrovaném obvodu 7432).

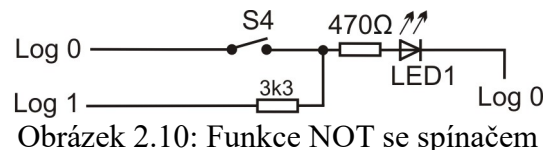


Obrázek 2.9: Funkce OR sestavená z funkce NAND

Úloha 2.3 Zapojení: 122 - log1, 122 - 125, 123 - 72, 72 - 71, 73 - 68, 126 - 66, 66 - 65, 67 - 69, 70 - 93, 92 - minus (log0), 123 - 112, 126 - 113, napájení 7400 - plus (log1), propojku rezistorů zapojit na minus.

Stiskem spínačů S3 a S4 měníte hodnoty vstupů. Sledujte, jak se obvod chová. Porovnejte chování s obvodem AND z obrázku 2.6.

Další logická funkce, se kterou se seznámíme, je funkce **NOT**. Ta pro nás bude mít význam „opačně“. Zapojíme ji opět nejdříve pomocí spínačů. Vymyslete takový obvod, který bude obsahovat spínač a ledku, ale ledka bude svítit tentokrát při **nestisknutém** spínači. Řešení je na obrázku 2.10.



Obrázek 2.10: Funkce NOT se spínačem

Úloha 2.4: Zapojení 10 - 126, 126 - 93, 92 - log0, 125 - log0, 9 - log1

Když spínač nezmáčkne, proud obvodem protéká přes rezistor 3k3, ochranný rezistor 470R a LED1. Když bude spínač sepnutý, proud poteče přes odpor 3k3 a přes sepnutý spínač k zemi. Přes ledku nepoteče nic. Splnili jsme zadání, ledka svítí, když spínač sepnutý není.

Pravdivostní tabulka funkce NOT je na obrázku 2.11.

| S4 | LED1 |
|----|------|
| 0 | 1 |
| 1 | 0 |



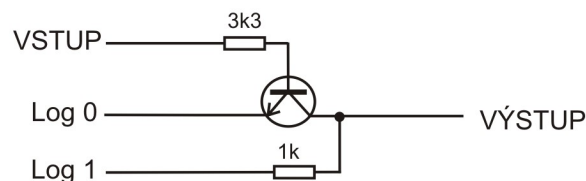
Obrázek 2.11: Pravdivostní tabulka funkce NOT a její schematická značka

Funkce, která „obrací“ log1 na log0, nebo obráceně log0 na log1, se nazývá **logická funkce NOT** (neboli **negace**).

Seznámili jsme se tedy s logickými funkcemi AND, OR a NOT. Zapořili jsme je pomocí spínačů a vše fungovalo dobře. Jak to ale funguje uvnitř integrovaného obvodu, kde je asi jasné, že spínače nenajdeme?

Nastavování hodnot výstupů se řeší součástkou, se kterou jsme se seznámili v předchozím díle návodu. Měla mimo jiné i funkci spínače. Vzpomínáte, jak jsme používali tranzistor jako spínač? No a v počítači jsou obvody zapojené pomocí tranzistorů, které tam nahrazují mechanické spínače.

Jak se nahradí mechanický spínač tranzistorem? Začnu s nejjednodušší funkcí NOT. Schéma zapojení je na obrázku 2.12.



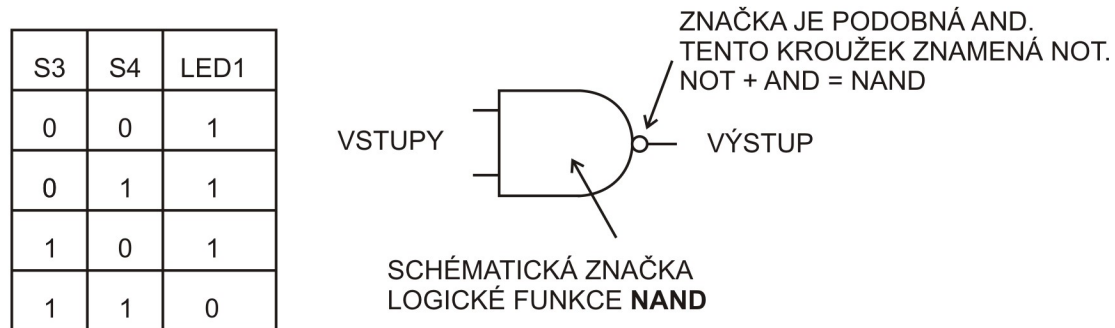
Obrázek 2.12: Funkce NOT s tranzistorem

Úloha 2.5: Zapojení: *VSTUP* je na zdířce 7, 8 - 60, 61 - log0, 59 - 6, 5 - log1, 59 je *VÝSTUP*

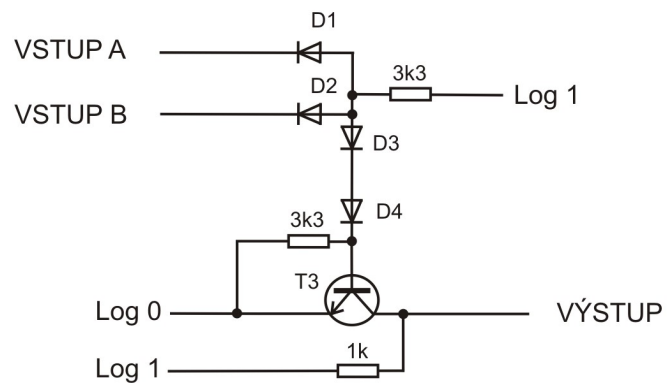
Když na vstup přivedeme log1, do báze poteče elektrický proud přes rezistor 3k3 a tranzistor se otevře. Následně poteče proud mezi kolektorem a emitorem, přes rezistor 1k rovnou na minus. Na kolektoru, a tím pádem na výstupu bude hodnota log0. Pokud přivedeme na vstup log0, tranzistor sepnutý nebude a na výstupu bude log1, protože proud poteče přes rezistor 1k.

Funkce AND a OR není tak snadné zapojit pomocí tranzistorů. Ukážeme si jinou funkci, kterou je jednoduché zapojit. Bude to negovaná funkce AND, tedy NOT AND, zkráceně se to píše **NAND**.

Jak vypadá pravdivostní tabulka funkce AND, to víme z tabulky na obrázku 2.2. Stručně - když jsou zmáčkнутé oba spínače, ledka svítí, jinak ne. Pravdivostní tabulka funkce NAND bude vypadat podobně, ale výstup bude obráceně. Tedy ledka **nebude** svítit jen v případě, že jsou všechna tlačítka stisknutá.



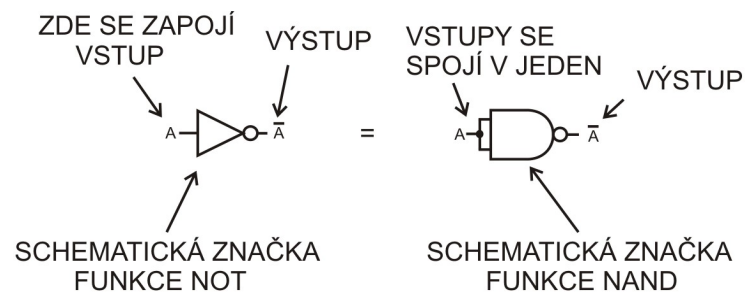
Obrázek 2.13: Tabulka funkce NAND a její schematická značka.



Obrázek 2.14: Schéma funkce NAND pomocí tranzistoru a diod

Obvod funkce NAND je velmi jednoduchý. Navíc má jednu vlastnost, kterou AND, OR ani NOT nemají. Pomocí dostatečného počtu funkcí NAND můžeme sestavit jakoukoliv logickou funkci. My jsme již sestavili funkci AND a OR. Sestavíme ještě funkci NOT a ještě jednou si ukážeme vše detailněji.

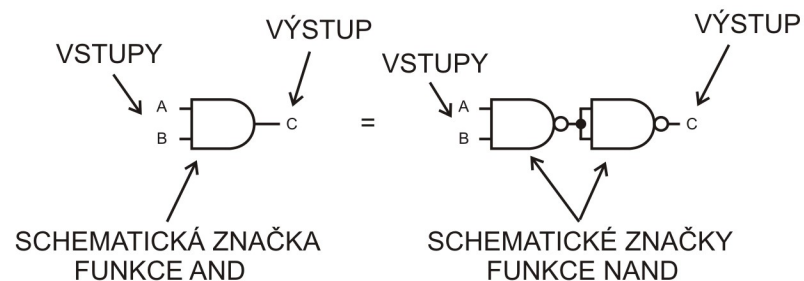
Začneme s nejjednodušší funkcí, a to je NOT. Pravdivostní tabulka je na obrázku 2.13. Sestavení funkce NOT pomocí jedné funkce NAND je jednoduché. NOT má pouze jeden vstup a jeden výstup. Spojíme vstupy NANDu a je to hotové. Schéma je na obrázku 2.15.



Obrázek 2.15: Zapojení funkce NOT pomocí funkce NAND

Obvod vlevo má stejnou funkci jako obvod vpravo. A to je negace, NOT.

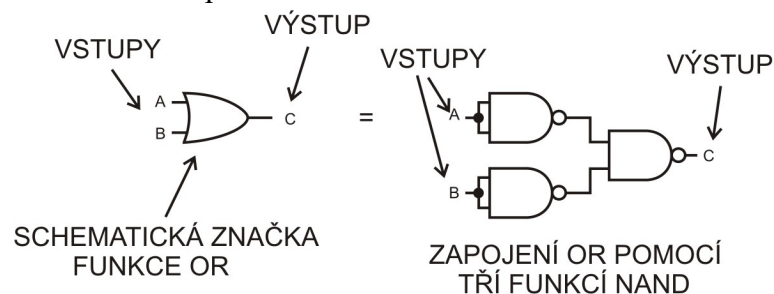
Kdybychom chtěli udělat z funkce NAND funkci AND, tak by musel být výstup funkce v pravdivostní tabulce na obrázku 2.13 „obráceně“, místo nuly jednička a místo jedniček nuly. To už umíme – použijeme na výstupu funkci NOT. Když víme, jak se NOT pomocí NAND vytvoří, je to jednoduché. Schéma obvodu je na obrázku 2.18.



Obrázek 2.18: Schéma sestavení funkce AND pomocí dvou funkcí NAND

Vstupy zde značím písmeny A a B, výstup písmenem C. Pokud budete hledat na internetu, můžete nalézt jiné značení. Vstupy se také značí X1, X2 a výstup písmenem Y.

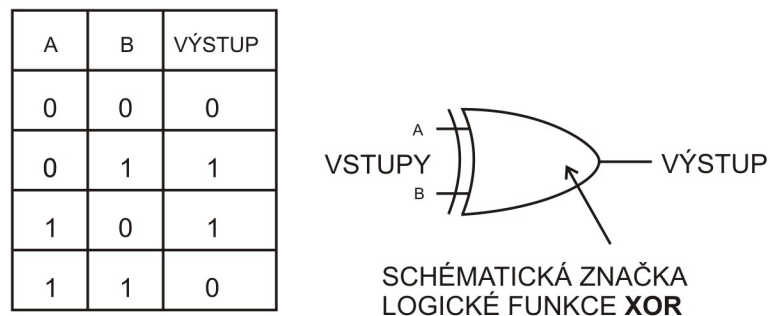
Teď vytvoříme funkci OR pomocí tří funkcí NAND. Když se pozorně zadíváte na pravdivostní tabulku na obrázku 2.15, tak si možná všimnete, že funkce NAND je vlastně OR pro logické nuly na vstupech. Stačí, aby alespoň jeden ze vstupů měl hodnotu log0, a výstup bude mít hodnotu log1. Toho se dá využít. Funkce OR má tu vlastnost, kdy stačí, aby alespoň jeden vstup měl hodnotu log1 a výstup bude mít hodnotu log1 také. Co jsem to vlastně udělal ve schématu na obrázku 2.19, nechám na důvtipu čtenáře.



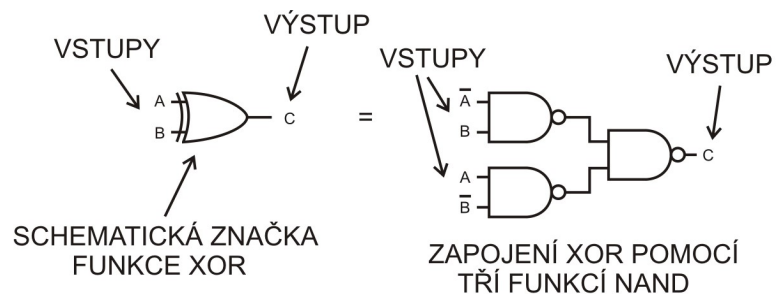
Obrázek 2.19: Schéma sestavení funkce OR pomocí tří funkcí NAND

Jak vidíte, funkce OR je náročnější na zapojení. Aby se vám logické funkce dobře sestavovaly, jsou výše uvedené schémata v jednodušší formě uvedené přímo na desce stavebnice, nad obvodem IO-2 7400.

Ještě pro úplnost uvedu logickou funkci **XOR**, která se nám bude také hodit. Je to vylučovací OR. Výstup nabývá hodnoty log1 pouze v případě, že hodnotu log1 má **pouze jeden** ze vstupů. Pravdivostní tabulka a schematická značka je na obrázku 2.20, její sestavení pomocí funkcí NAND je na obrázku 2.21.



Obrázek 2.20: Pravdivostní tabulka funkce XOR a její schematická značka



Obrázek 2.21: Schéma sestavení funkce XOR pomocí tří funkcí NAND

Kapitola 3 - Sčítačka binárních čísel

Tato kapitola je dobrovolná, je určena pro ty, kteří se chtějí o binárních číslech dozvědět ještě něco více. Potom je sčítačka obvod, který zde nemůže chybět. Už víme, že čísla jsou v počítači uložena jako logické nuly a jedničky, kterým říkáme bity. Jak ale počítač dvě čísla sečte? Jednat se bude zatím jen o sčítání dvou bitů, nikoliv opravdu celých čísel.

Nejdříve si popíšeme, jaké jsou vlastně výsledky sčítání. Sčítání „obyčejných“ jedniček a nul z běžného života jistě znáte. $1+1=2$. Já zde sepíšu všechny kombinace, které nám mohou nastat při sčítání dvou bitů.

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 2$$

V první kapitole jsme si ukázali, jak vypadá binární zápis čísla. Binární zápis čísla 2 je „10“. Nechte se to „deset“ jak jsme zvyklí z desítkové soustavy, ale „jedna nula“. Zápis kombinací nyní provedu pomocí \log_0 a \log_1 .

$$\text{Log}0 + \text{log}0 = \text{log}0$$

$$\text{Log}0 + \text{log}1 = \text{log}1$$

$$\text{Log}1 + \text{log}0 = \text{log}1$$

$$\text{Log}1 + \text{log}1 = \text{„10“}$$

Poslední řádek je asi matoucí. My už víme, že každá logická funkce může mít více vstupů, ale výstup má vždy jen jeden. Jak tedy vypadá logická funkce, která bude tvořit sčítačku, když jeden z výsledků není jen 1 nebo 0, ale 10? Řešení je vlastně jednodušší, než by se zdálo. Vytvoříme funkce dvě. Zápis kombinací sčítačky provedu ještě jednou, tentokrát výsledky zapíšu dvouciferně.

$$0 + 0 = 00$$

$$0 + 1 = 01$$

$$1 + 0 = 01$$

$$1 + 1 = 10$$

A celé to sepíšeme do tabulky

| VSTUPY | | VÝSTUPY | |
|--------|---|---------|----|
| A | B | Y1 | Y2 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Obrázek 3.1: Sčítání binárních čísel

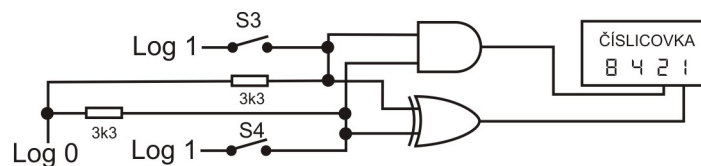
Na obrázku 3.1 vidíme, že vstupy jsou dva, A a B. Výstupy ale nejsou dva. Vytvoříme totiž dvě různé logické funkce. Obě budou mít stejné vstupy, každá bude mít jen jeden výstup. Ve sloupci Y1 je logická funkce pro **přenos** do vyššího řádu, ve sloupci Y2 je funkce pro **součet**. Pro každou z těchto funkcí sestavíme obvod.

Funkce Y1 - takovou tabulku už známe. Nabývá hodnoty 1 jen v případě, že oba vstupy jsou 1. To je AND. Funkce Y2 - takovou tabulku už známe taky. To je XOR. Funkce zapíšeme takto:

$$Y1 = A \text{ AND } B$$

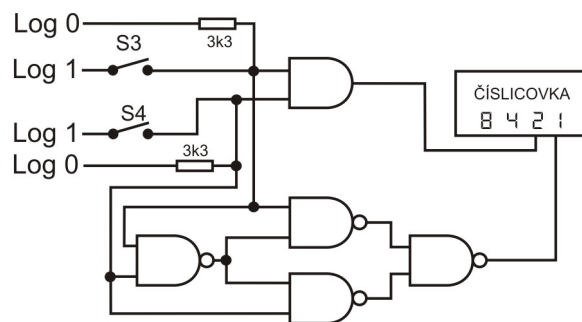
$$Y2 = A \text{ XOR } B$$

A sestavíme obvod, kterému se říká **poloviční sčítačka**. Schéma je na obrázku 3.2.



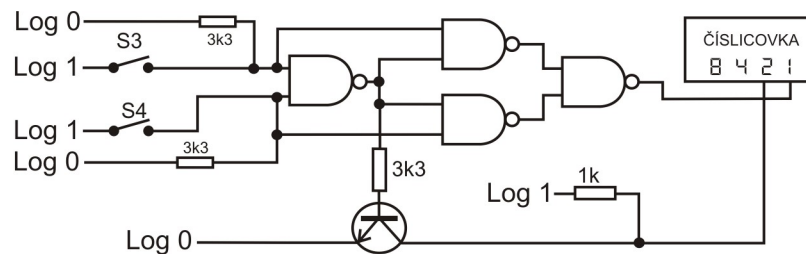
Obrázek 3.2: Obvod poloviční sčítačky

Pokud obvod 7486 nemáme, musíme ještě sestavit funkci XOR pomocí funkce NAND. Zapojení je na obrázku 3.3.



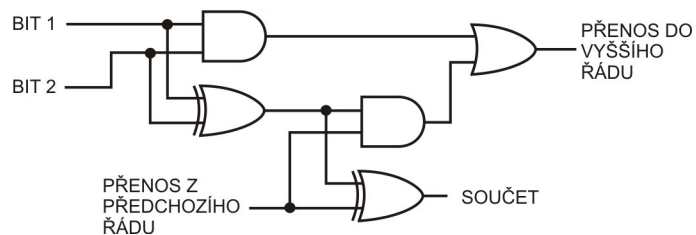
Obrázek 3.3: Obvod poloviční sčítačky pomocí NAND a AND

Pokud nemáme k dispozici Saimon 2 (tím pádem obvod AND) a chceme to celé zapojit na Saimon 1, bude nám scházet jedna funkce NOT. Tu vytvoříme pomocí tranzistoru podle kapitoly 2, obrázku 2.14. Výsledný obvod je na obrázku 3.4.



Obrázek 3.4: Obvod poloviční sčítačky na Saimon 1

Možná vás zajímá, jak vypadá **úplná sčítačka**. Naše poloviční sčítačka měla dva vstupy, dva sčítané bity. Výstupy byly dva - **součet** a **přenos** do vyššího řádu. Úplná sčítačka má vstupy tři, tím třetím je přenos z předchozího sčítání. Obvod lze realizovat pomocí dvou polovičních sčítaček a funkce OR. Schéma je na obrázku 3.5.



Obrázek 3.5: Úplná sčítačka

Ted' si ukážeme součet na příkladu. Sečteme dvě čísla, $4 + 6$. Jejich binární zápis je $100 + 110$. Napíšeme si čísla pod sebe.

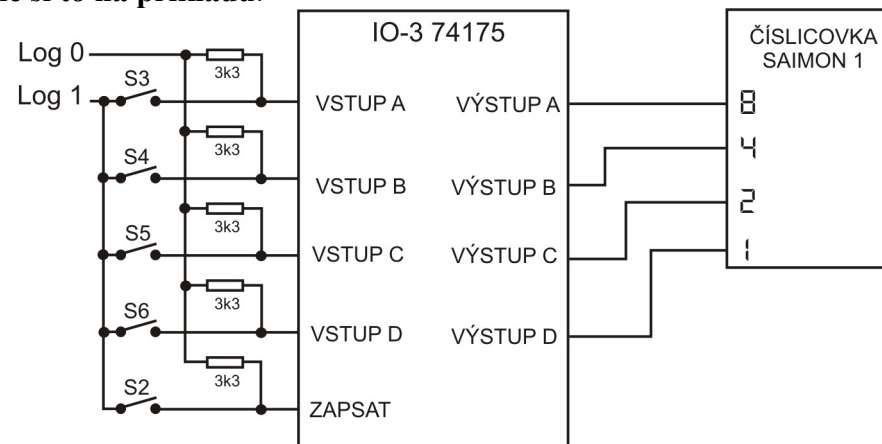
$$\begin{array}{r} 100 \\ 110 \\ \hline \end{array}$$

Na takový součet potřebujeme tři úplné sčítačky. Každý sloupec pro nás představuje jeden řád. Každá sčítačka sečte bity stejného řádu. Sečteme $0+0 = 0$, přenos 0. Dále sčítáme následovně - sečteme bity druhého řádu a přenos z předchozího $0+1+0 = 1$, přenos do dalšího řádu je 0. Sečteme třetí řád a přenos z předchozího řádu $1+1+0 = 0$ s přenosem 1. Výsledek (poslední číslo, které nám vyšlo, píšeme vlevo) je **1010**. Je to stejné sčítání „pod sebou“ ze základní školy.

Kapitola 4 - Elektronická paměť

Poslední integrovaný obvod má označení IO-3 74175 4xD-KO. U tohoto integrovaného obvodu můžete vidět více vstupů a dokonce více výstupů. To je v pořádku, nejedná se o logickou funkci, ale o **paměť**. Obvody logických funkcí mají výstup jeden. Paměťový obvod může mít výstupů více.

Jak funguje paměť, konkrétně integrovaný obvod 74175? Pokud vstupy obvodu (které jsou označeny písmeny ABCD) nastavíme na určitou hodnotu (log0 nebo log1), přivedeme na krátkou dobu hodnotu log1 na zdířku označenou ZAPSAT, tak obvod nastaví výstupy na hodnotu, která byla na vstupu se stejným písmenkem. **Ukážeme si to na příkladu.**



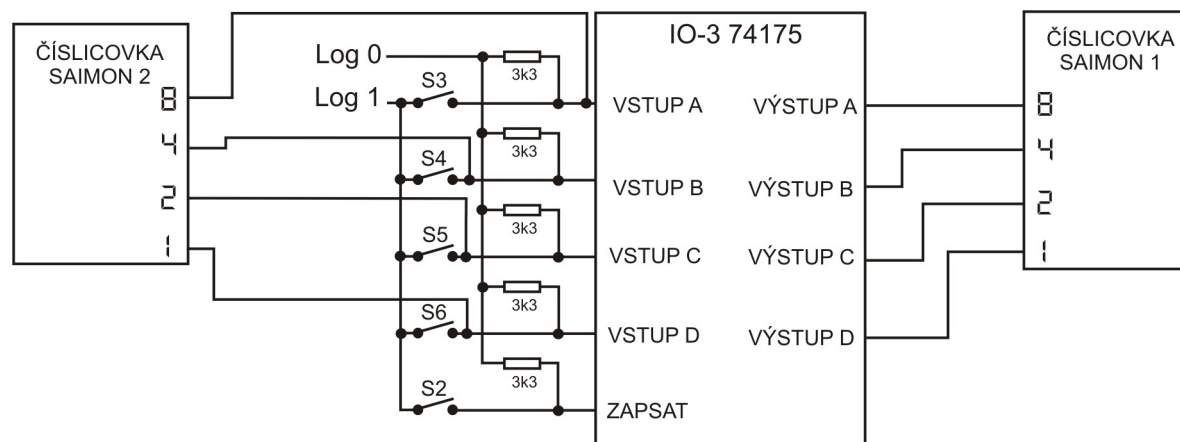
Obrázek 4.1: Obvod 74175 - paměť

Úloha 9: Zapojení 119 – plus, 119 - 122, 122 – 125, 125 – 128, 128 – 131, 120 – 87, 123 – 77, 126 – 76, 129 – 83, 132 – 84, 79 – 88, 74 – 89, 81 – 90, 86 – 91, 112 – 123, 113 – 126, 114 – 129, 115 – 132, napájení číslicovky – plus, napájení 74175 – plus, propojku rezistorů na minus.

Číslo, které nastavíte kombinací sepnutých spínačů S3 až S6, obvod 74175 zapíše po stisku spínače S2. Zapsané číslo se zobrazí na číslicovce. Můžete zapínat a vypínat spínače S3 – S6, ale zapsané číslo se nezmění.

Obvodu 74175 se říká registr, nebo **klopný obvod**. To znamená, že nastaví (neboli naklopí), své výstupy podle hodnoty, kterou jsme přivedli na odpovídající vstup. Zápis proběhne při stisku spínače S2, tedy ve chvíli, kdy změníme log0 na log1 na vývodu označeném jako ZAPSAT (v anglické literatuře najdete označení **CLOCK**, hodiny). Zápis se děje jen při **změně z log0 na log1**. Tomu říkáme **náběžná hrana**, nebo **hodinový pulz**. I když na vývodu ZAPSAT zůstane log1 nebo log0 trvale, tak se výstupy dále nezmění.

Pokud máme k dispozici Saimon 2, zapojení vylepšíme. Zobrazíme jak zapisované číslo, tak již zapsané. Přidáme číslicovku na Saimon 2.



Obrázek 4.2: Obvod 74175 – paměť, zobrazení obou čísel

Druhou číslicovku připojíme na vstupy. Abychom opět nemuseli používat dlouhé drátky, použijeme propojky. Nikoliv však propojky X, ale propojky vstupů a výstupů obvodu 74175. Ty jsou uprostřed modulu Saimon 2, viz obrázek 4.3.

| VSTUPY | | VÝSTUPY | | | |
|--------|--|---------|--|--------------|--|
| 77 A | | 79 A | | 78 \bar{A} | |
| 76 B | | 74 B | | 75 \bar{B} | |
| 83 C | | 81 C | | 82 \bar{C} | |
| 84 D | | 86 D | | 85 \bar{D} | |

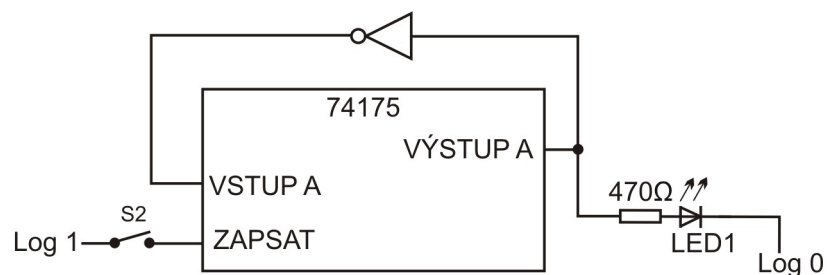
Obrázek 4.3: Propojky vstupů a výstupů obvodu 74175

Rozdíl mezi propojkami X je ten, že tyto propojky jsou vždycky připojené na vstupy a výstupy IO-3 74175. Takže pokud číslicovku na Saimon 2 připojíme na vstupy ABCD v tomto poli, nemusíme na Saimon 1 nic dalšího zapojovat. Jednoduše přidáme k úloze 9 následující zapojení.

Úloha 9: Zapojení rozšíření 259 – 77 A, 260 – 76 B, 261 – 83 C, 262 – 84 D, napájení číslicovky – plus

Obvod 74175 má tedy vstupy A, B, C, D, a výstupy A, B, C, D. K čemu jsou, to už teď víme. Když na nějaký vstup, třeba VSTUP A, přivedeme log1, na zdiřku ZAPSAT přivedeme náběžnou hranu, tak na výstupu se stejným písmenkem, tedy VÝSTUP A, se objeví také log1. Má ale ještě výstupy, které mají nad písmenkem čáru. Ta čára znamená, že ten výstup má obrácenou hodnotu oproti výstupu bez čáry. Když bude na zdiřce VÝSTUP A hodnota log1, tak na výstupu \overline{A} bude hodnota log0. Je to vlastně vývod výstupu A, za kterým je ještě zapojená funkce NOT. To je moc důležité pro konstrukci obvodů, které se naučíme ve třetím díle návodu.

Malý úvod, jaké obvody to budou, udělám už teď. Zkuste zapojit následující schéma.

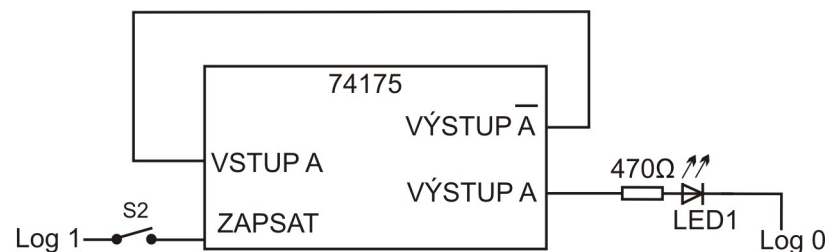


Obrázek 4.4: Obvod 74175 jako jednoduchý sekvenční obvod

Úloha 10: Zapojení 71 - 72, 71 - 79, 73 - 77, napájení 7400 - plus, napájení 74175 - plus, plus - 119, 79 - 93, 120 - 87, 92 - minus

Když zapojíme obvod z obrázku 4.4, tak bude mít velmi zajímavou funkci. Po stisku S2 si zapamatuje úroveň, která je na vstupu A. Tam ale bude opačná hodnota, než je na výstupu A, protože funkce NOT nám ji otočí. Když bude na výstupu A hodnota log0, tak si obvod zapamatuje hodnotu log1, protože bude mít na vstupu A opačnou hodnotu díky funkci NOT. Po stisku S2 se nastaví log1 na výstupu A. Funkce NOT nám ji ale zase otočí na log0 a přivede na vstup. A po dalším stisku S2 si obvod zapamatuje log0, kterou nastaví na výstupu A. A tak dále. Mačkáním S2 by se měla na výstupu pravidelně měnit hodnota z log0 na log1 a obráceně.

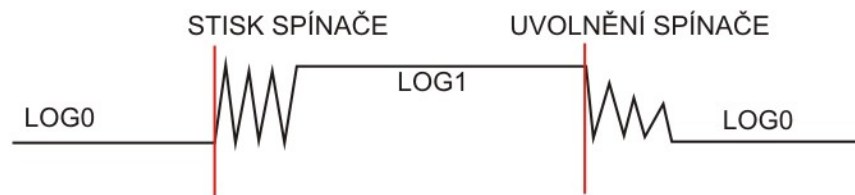
IO 74175 má v sobě zabudovanou funkci NOT pro výstupy. Jsou to ty výstupy s čarou, které mají vždy opačnou hodnotu, než výstup se stejným písmenkem. Obvod na obrázku 4.5 má stejnou funkci, jako obvod na obrázku 4.4. Dobře si je oba prohlédněte, tentokrát nepoužíváme IO 7400.



Obrázek 4.5: Obvod 74175 jako jednoduchý sekvenční obvod

Stiskem spínače S2 zjistíte, že se hodnota na výstupu mění nepřesně. LED1 bliká nepravidelně, rozhodně se nedá mluvit o tom, že by se pravidelně střídala 0 a 1. Stisknutím spínače S2 se totiž nezmění úroveň z log0 na log1 okamžitě. Chvilí to jakoby „jiskří“, rychle se to mění z log0 na log 1 a obráceně a až pak se hodnota ustálí. To jiskření se děje tak rychle, že to lidské oko není schopné vidět, ale integrovaný obvod je velmi citlivý - „vidí to“. Když spínač uvolníte, děje se to samé. Zase chvíli trvá, než se z log1 stane log0. Obvod 74175 každou změnu bere jako povel zapsat.

Spínač je pro získání „kvalitní“ náběžné hrany nevhodný. Potřebujeme nějaký lepší zdroj náběžných hran, kde změna proběhne jen jednou. Obrázek 4.6 ukazuje představu, jak to "jiskření" u spínače asi probíhá. Tomu jiskření říkáme zákmit nebo **přechodový jev**.

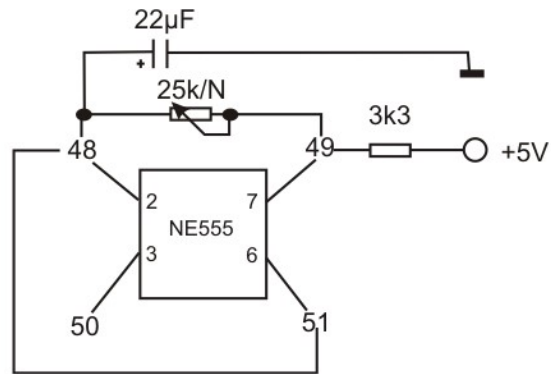


Obrázek 4.6: Zákmit při stisknutí spínače.

Kapitola 5 - Obvody pro získání hodinových pulzů

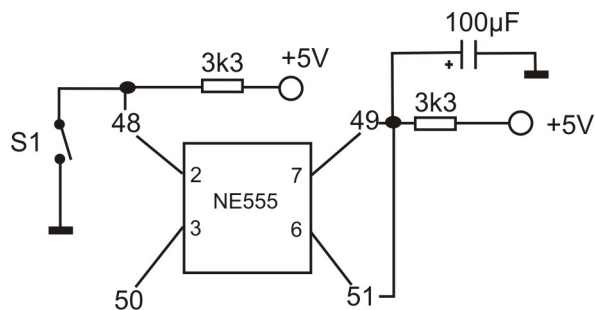
Nejlepší zdroj náběžné hrany (hodinového pulsu) je integrovaný obvod NE555. Jeho podstatné funkce známe z prvního dílu Saimon 1. Já je zde zopakují.

Astabilní obvod – na výstupu (zdička 50) se stále mění hodnota z log0 na log1 a zase zpět. Neustále tedy generuje hodinové pulsy (náběžné hrany). Rychlost jsme určovali hodnotou kondenzátoru v obvodu a hodnotou odporu potenciometru. Obvod je na obrázku 5.1.



Obrázek 5.1 - Astabilní zapojení obvodu NE555

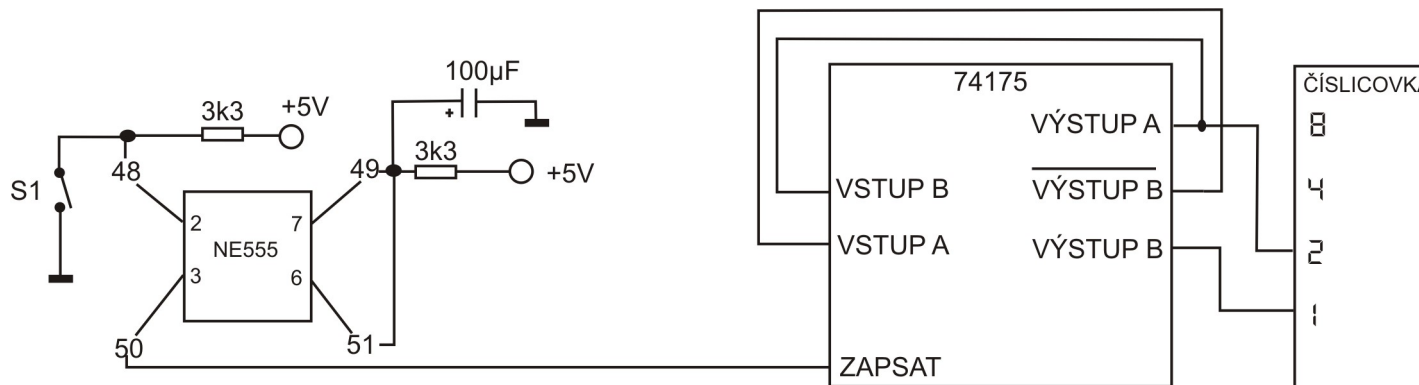
Monostabilní obvod – na výstupu je hodnota log0. Při stisku spínače se na určitou dobu hodnota výstupu změní na log1. Tím se vygeneruje hodinový puls. Po uplynutí času se změní zpět na log0. Schéma obvodu je na obrázku 5.2.



Obrázek 5.2 – Monostabilní zapojení obvodu NE555

Tyto dvě schémata pro získání kvalitního hodinového pulsu pro nás budou podstatná. Existují další zdroje hodinových pulsů, které uvedu na konci návodu jen pro doplnění.

Zapojíme teď sekvenční obvod s IO 74175, který má přiveden na zdiřku ZAPSAT hodinový puls z obvodu NE555. Vybereme nejdříve monostabilní zapojení.



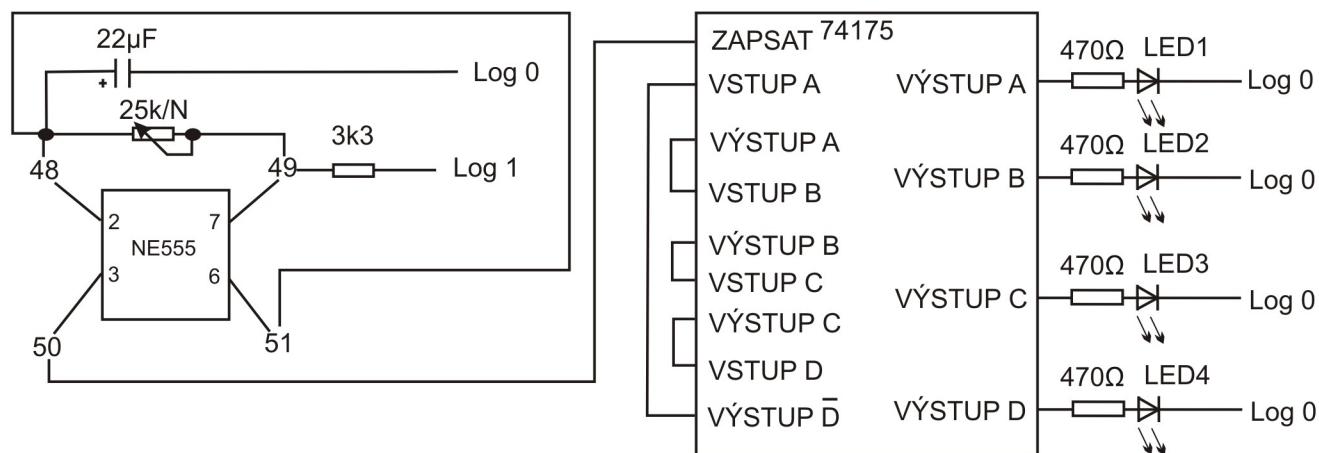
Obrázek 5.3 - Obvod 74175 jako jednoduchý sekvenční obvod s generátorem hodinových pulzů

Úloha 11: Zapojení: 10 - 51, 9 - log1, 8 - 48, 7 - log1, 49 - 51, 49 - 43, 48 - 114, 116 - log0, 77 - 78, 79 - 93, 92 - log0, 42 - log0, 50 - 87
Zapnout IO 555 a IO74175.

Obvod na obrázku 5.3 už bude fungovat perfektně. Stiskem spínače se bude hodnota výstupu krásně měnit. Pokud místo monostabilního zapojení NE555 jako zdroje hodinových pulzů zapojíte astabilní zapojení, bude to blikat. Ve třetím díle návodu si ukážeme, k čemu je to dobré a budeme stavět velmi jednoduchý „počítač“, respektive automat.

Pro pochopení účelu stavebnice Saimon je touto kapitolou řečeno vše podstatné a můžete přejít k třetímu dílu návodu. Já zde uvedu ještě několik zapojení, které se mi velmi líbily, jako třeba světelný had. Dále si ukážeme další obvody pro získání hodinových pulzů, ukážeme si, co to je klopný obvod a jak funguje, jak si zapojit na stavebnici jedno-bitovou paměť.

Obvod, který se mně osobně moc líbí, využívá vlastnosti obvodu v úloze 11. Zapojíme ale všechny vstupy a výstupy. Výsledkem bude krásný světelný „had“. Funkci obvodu zkuste odhadnout.



Obrázek 5.4 – Světelný had

Úloha 12: Zapojení:

časovač: 49 - 10, 9 - log1, 48 - 51, 23 - 48, 24 - 49, 48 - 39, 38 - log0, 50 - 87

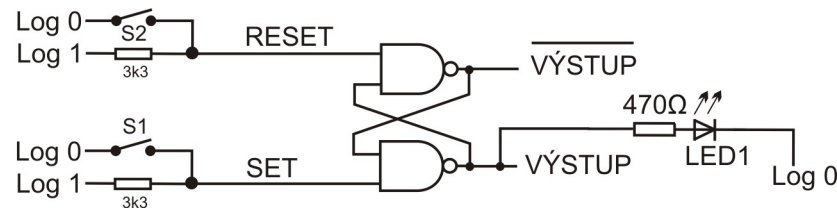
integrováný obvod: 77 - 85, 79 - 76, 74 - 83, 81 - 84

LEDky: 79 - 93, 74 - 95, 81 - 97, 86 - 99, 92 - 94, 94 - 96, 96 - 98, 98 - log0

Pokud máme Saimon 2, zapojte ještě 79 – T1, 78 – T5, 74 – T2, 75 – T6, 81 – T3, 82 – T7, 86 – T4, 85 – T8

Kapitola 6 - Další tvarovací obvody

Vynikající, a přitom jednoduchý obvod pro kvalitní signál změny log0 na log1 a obráceně, se jmenuje RS. Znamená to Reset - Set, česky něco jako Vynuluj - Nastav. Jeho schéma je na obrázku 6.1.

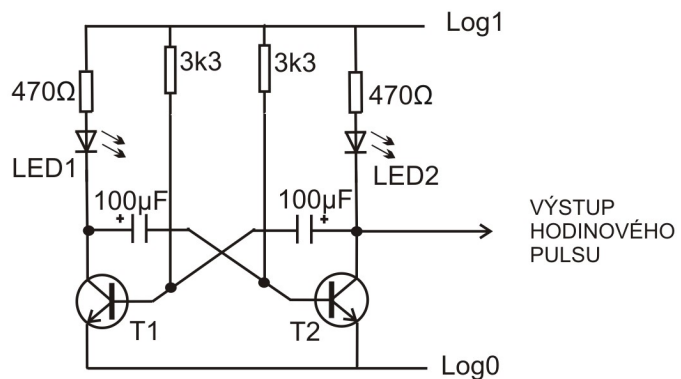


Obrázek 6.1: Obvod RS

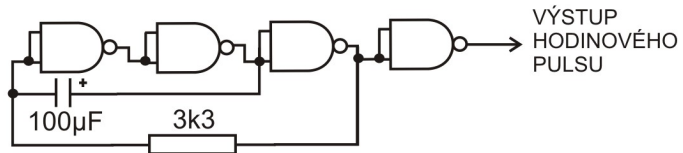
Úloha 13: Zapojení 116 – minus, 117 - 71, 110 - 117, 119 - minus, 120 – 111, 73 - 69, 72 - 70, 68 - 120, 73 - 93, 92 – minus, Zapnout IO-2 7400, propojku rezistorů na plus

Spínačem S1 obvod natrvalo nastavíme na log1 na výstupu. Jakékoliv zakmitání nemá na výstup vliv, protože obvod se pomocí S1 dá jen nastavit, vynulovat už ale ne. Pro nastavení log0 na výstupu slouží spínač S2. Který umí obvod zase jen vynulovat, takže zákmity nám také nevadí. Výstup je označen dole, na tom spodním NANDu. Na tom horním je výstup komplementární (zkuste si na něj přivést třeba LED2, bude svítit opačně k LED1). A ještě poznámka - kdo si všiml, tak to nastavení provádíme logickými nulami, čili po stisknutí například S1 se na vstup, který je označený SET přivede log0 a obvod nastaví na výstupu log1.

Dál chci uvést, že jako zdroj hodinového signálu se dají použít i blikáče a majáky z prvního dílu návodu. Uvedu zde jen schémata, kde bude označen výstup hodinového signálu.



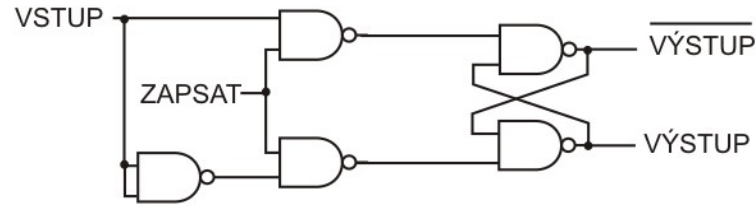
Obrázek 6.3: Tranzistorový blikáč jako zdroj hodinového pulsu



Obrázek 6.4: Blikáč s obvodem 7400 jako zdroj hodinového pulsu

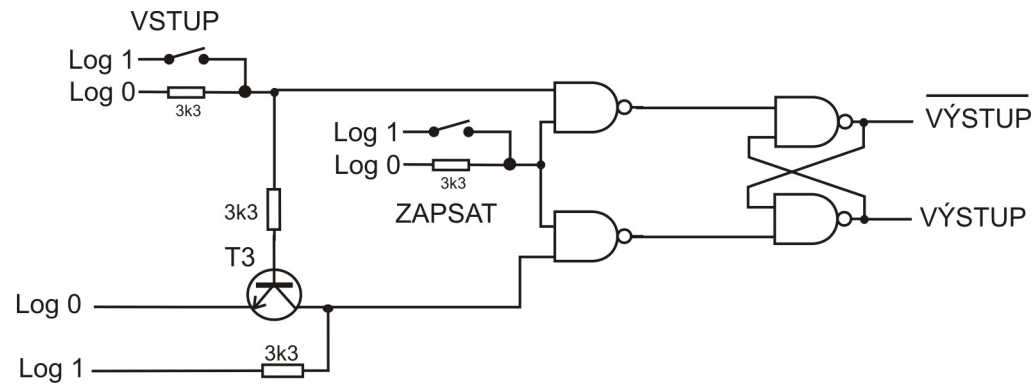
Kapitola 7 - Klopný obvod typu D

Zde chci ukázat, co je vlastně uvnitř IO 74175 a jak si zapojit jeden vstup-výstup (paměťovou buňku) pomocí funkcí NAND. Ve schématu je obvod SR, který už známe, ale je zapojený ještě s dalšími NANDy. Takovému obvodu se říká **D-klopný obvod**. Opět jeden obrázek za tisíc slov, nuže hleďte na obrázek 7.1.



Obrázek 7.1: D-Klopný obvod

Kdo by si to chtěl zapojit, tak si musíme trochu pomoci, protože NANDy máme na stavebnici jen čtyři, ale zde jich je potřeba pět. Nejlepší bude NAND vlevo, který plní funkci NOT, nahradit obvodem s tranzistorem (obrázek 2.14). Můžete si to zkusit zapojit a vyzkoušet.



Obrázek 6.2: D-Klopný obvod s tranzistorem

Teď je i jasné, proč má IO74175 i opačné výstupy, protože v tom obvodu nám prostě vznikly a je jednoduché je vyvést ven.

Obsah

| | |
|--|----|
| Úvod..... | 2 |
| Připojení Saimon 2..... | 3 |
| Kapitola 1 - Jak počítač „vidí“ čísla | 4 |
| Kapitola 2 - Logické funkce | 10 |
| Kapitola 3 - Sčítačka binárních čísel | 21 |
| Kapitola 4 - Elektronická paměť..... | 25 |
| Kapitola 5 - Obvody pro získání hodinových pulzů | 29 |
| Kapitola 6 - Další tvarovací obvody | 33 |
| Kapitola 7 - Klopný obvod typu D | 35 |
| Slovník pojmů..... | 37 |

Slovník pojmů

| | |
|--|---|
| Logická 0, Log.0: | Stav vypnuto, nesvítí, není tam napětí. |
| Logická 1, Log.1: | Stav zapnuto, svítí, je tam napětí. |
| Dvojková, binární soustava čísel: | Počítačový způsob zobrazení čísel pomocí nul a jedniček |
| Šesnáctková, hexadecimální soustava čísel: | Desítková soustava čísel rozšířená o znaky A,B,C,D,E,F. |
| Desítková, decimální soustava čísel: | Soustava vymyšlená pro člověka – má deset prstů, tak se dobře počítá od jedné do deseti. |
| Bit (angl. BInary digiT) | Nejmenší jednotka zápisu čísla, znamená zapnuto, vypnuto, tvoří ji log 1 nebo log 0 |
| Logická funkce: | Pro nás obvod, který má jeden nebo více vstupů a jeden výstup. Nastavuje hodnotu výstupu podle hodnot vstupů podle příslušné pravdivostní tabulky |
| Vstup logické funkce | Místo v obvodu, kam přivádíme hodnotu log0 nebo log1, například pomocí spínačů. |
| Výstup logické funkce | Místo v obvodu, kde se nastavuje odpovídající logická hodnota podle hodnot vstupů. |
| Logická funkce Chyba! Nenalezen zdroj odkazů.: vstupy mají hodnotu log 1 | Logický součin – výstup nabývá hodnoty log1 pouze pokud oba |
| Logická funkce OR: | Logický součet – výstup nabývá hodnoty log1 pokud jeden, nebo oba vstupy mají hodnotu log 1 |
| Logická funkce NOT, negace: | Logický opak – výstup nabývá opačné hodnoty, než na kterou je nastaven vstup |

| | |
|---------------------------------------|---|
| Logická funkce NAND : | Je to logická funkce podobná AND. Výstup je ale opačně. Tak, jako bychom za AND zapojili ještě logickou funkci NOT |
| Logická funkce XOR : | Vylučovací (exkluzivní) OR. Výstup nabývá hodnoty log1 pokud pouze jeden ze vstupů nabyl hodnoty log1 |
| pravdivostní tabulky : | Tabulky, v kterých jsou napsány hodnoty vstupů (log0 nebo log1) a odpovídající výstupy podle příslušné logické funkce, ke které je tabulka napsána. |
| TRUE : | Česky „pravda“, logická 1, stav napětí. |
| FALSE : | Česky „nepravda“, logická 0, stav bez napětí. |
| hodinový pulz, náběžná hrana : | Změna z hodnoty log0 na hodnotu log1. Musí proběhnout jen jednou. |