

# SALMON I PRO ŠKOLY

NÁVOD - ČÁST TŘETÍ  
STAVÍME MINIATURNÍ POČÍTAČ  
ANEB SEKVENČNÍ OBVODY  
A KONEČNÉ AUTOMATY

## **Kapitola 1 - Obvod, který „žije“ – co si pod tím představit**

V téhle části návodu se dostaneme k hlavní myšlence stovebnice. Ukážu vám ten hlavní nápad a také, co všechno nám to přináší. Takže tedy – obvod, který žije – to je můj výraz, asi ho nikde jinde nevidíte. Ale líbí se mi. Co to znamená? Vzpomeňte si na různé obvody z minulých dílů návodu. Obvody svítily. Dokonce i zhasínaly, když v obvodu byl spínač. A po seznámení s tranzistorem blikaly nebo bzučely. Díky tranzistorům dokázal obvod měnit svůj stav.

Co je stav obvodu? Stav je například „ledka svítí“. A změní se do stavu „ledka nesvítí“. To se děje automaticky, stále dokola. Obvod „žije“ - tím mám na mysli, že obvod sám přechází mezi stavy, které můžeme pozorovat.

Blikač sice žije, ale já vás zde chci naučit navrhovat a zapojovat obvody, které „žijí mnohem více“. Budou mít více stavů, a budeme moci ovládat, do kterého stavu obvod přejde. Pro ovládání bude mít obvod vstupy – například spínače. A pro zobrazení stavu výstupy – například ledky a číslicovku. Na výstupech budeme pozorovat „co obvod dělá“.

Bude to takový velmi jednoduchý „miniaturní počítač“, spíše automat, s napevno daným programem – ten budou tvořit logické funkce. Místo klávesnice budou spínače a místo monitoru budou ledky nebo číslicovka. Budu jim říkat **Synchronní sekvenční obvody**, zkráceně **SSO**.

První, a také hlavní část SSO bude integrovaný obvod 74175. Víme, že je to **paměťový registr**, který má čtyři vstupy, čtyři výstupy, čtyři negované výstupy. Také víme, že jeho funkce je nastavit výstupy podle hodnot vstupů ve chvíli, kdy se na vývodu ZAPSAT objeví hodinový pulz. Zapamatuje si stavy vstupů.

Druhá důležitá část SSO bude **zdroj hodinových pulsů**, který nám bude řídit změny stavů SSO. Bude napojen na zdířku ZAPSAT obvodu 74175. Jako zdroj hodinových pulsů si z těch všech možných z předchozího dílu návodu vybírám časovač NE555 - buď v astabilním, nebo monostabilním zapojení.

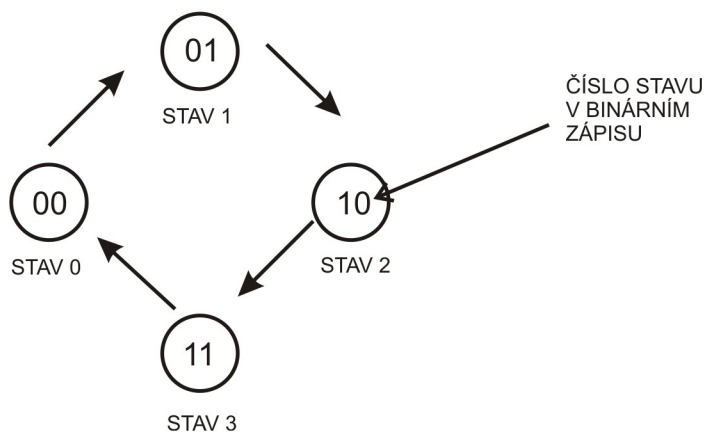
Tyhle obvody už známe. Než vysvětlím, jaké jsou další části, tak uvedu příklad jednoduchého SSO. Bude to obvod, který bude zobrazovat na číslicovce postupně čísla 0,1,2,3 a opakovat to stále dokolečka. Můžu vám teď tady obvod rovnou ukázat a napsat jeho zapojení. Já vás chci ale naučit takové obvody vymyslet. Návrh SSO si rozdělíme do několika kroků.

### Krok 1 – slovně si popíšeme, jaká bude funkce obvodu

Popis bude vypadat asi takto – obvod bude zobrazovat na číslicovce číslice 0,1,2,3 a pak zase 0,1,2,3 a stále dokola. Jsou potřeba čtyři různé stavy obvodu. Stav číslo 0, ve kterém obvod zobrazí číslici 0. Stav číslo 1, ve kterém obvod zobrazí číslici 1. Stejně tak stav 2 zobrazí číslici 2 a stav 3 zobrazí číslici 3. Pak obvod musí přejít zpět do stavu 0.

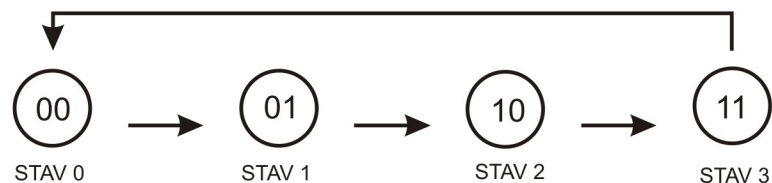
### Krok 2 – nakreslíme si, jak jdou stavy po sobě

Většinou budeme kreslit jakýsi graf. Kolečka, do kterých napíšeme čísla stavů SSO. Mezi kolečky nakreslíme šípky, abychom věděli, jak obvod mezi stavy prochází. Stavy budu **vždy číslovat v binární soustavě**. Čísla stavů tedy budou 00, 01, 10, 11. Kresba může vypadat takto:



Obrázek 1.1 – graf stavů počítadla

Nebo takto:



Obrázek 1.2 – Jiný graf stavů počítadla

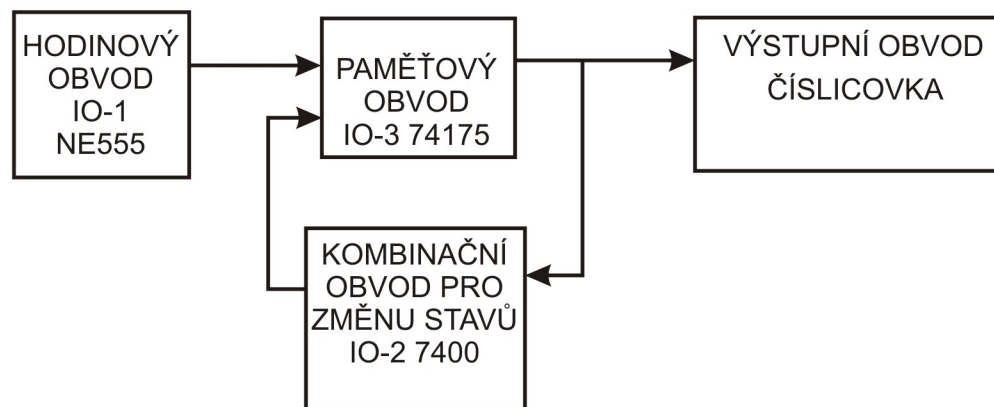
Oba grafy zobrazují to samé. Šipky ukazují, jak obvod přechází mezi stavy. Když máme graf nakreslený, můžeme přejít k třetímu kroku. K pochopení kroku 3 je ale potřeba vědět, co to je **blokové schéma** synchronního sekvenčního obvodu.

Schéma obvodu pro nás byl obrázek, kde byly nakreslené elektronické součástky a šipky, které nám říkaly, jak jsou součástky spolu propojené. **Blokové schéma** znamená, že už nekreslíme obvod jako celek se všemi jeho součástkami. Nakreslím jen obdélníky (bloky), do kterých napíšu, co blok (kousek obvodu) dělá a jak jsou spolu tyto kousky spojené.

Například nakreslím blok, do kterého napíšu „Zdroj hodinových pulzů“. Znamená to, že tím myslím jakýkoliv obvod (v minulém dílu jsme jich jmenovali mnoho), který bude mít výstupní svorku s hodinovými pulsy. Jak ale schéma obvodu vypadá, nás v tuto chvíli moc nezajímá. Může to být třeba obvod s časovačem NE555.

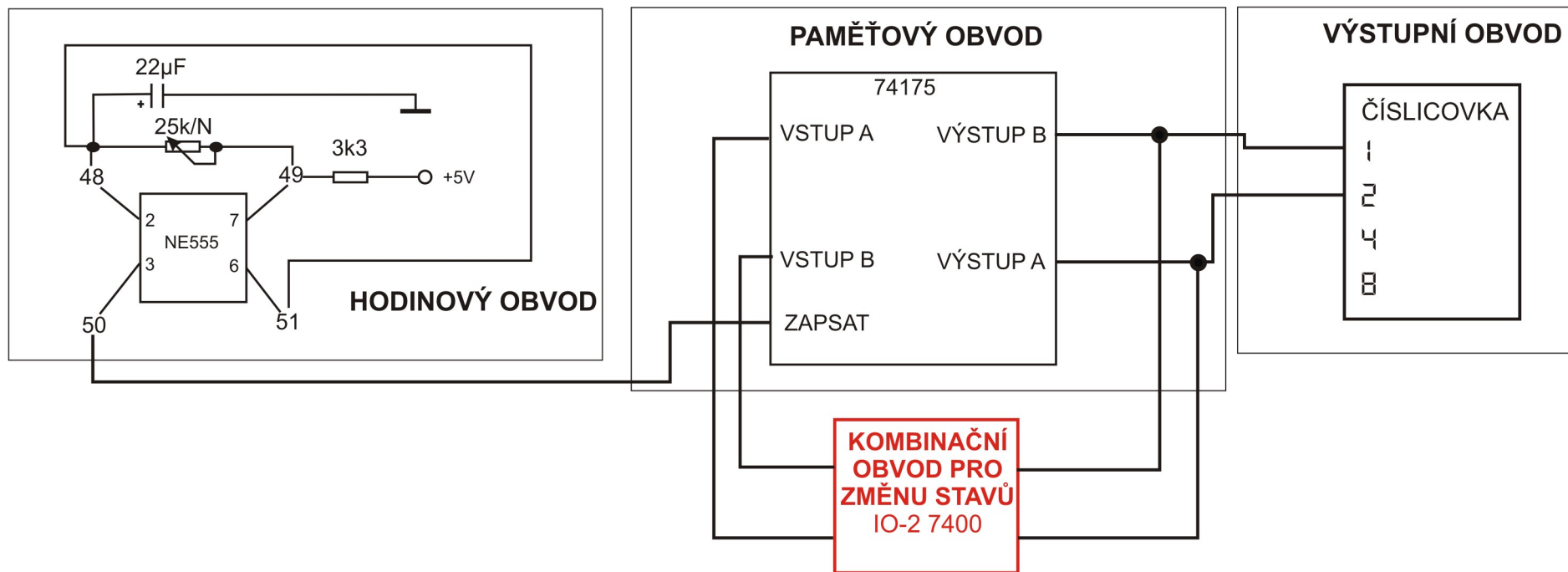
Poznámka - v předchozím dílu návodu jsem v úloze 12 rozdělil zapojení drátků na „časovač“, „integrováný obvod“ a „ledky“. To jsou vlastně tři bloky, které jsou spolu spojené.

Blokové schéma našeho počítačla vypadá takto:



Obrázek 1.3 – blokové schéma počítačla

Ted' schéma překreslím do podoby, kterou známe. Rámečkem označím jednotlivé bloky.

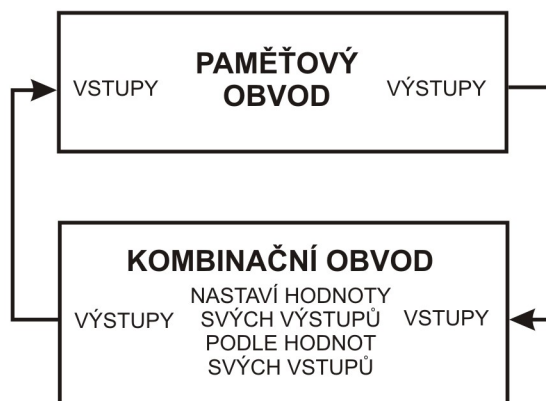


Obrázek 1.4 – Schéma počítadla

Vidíte, že každý blok obvodu má uvnitř schéma, které je nám známé. Jediný blok zůstal stále tajemný. A to je blok s názvem **kombinační obvod pro změnu stavů**. Ten se budeme učit navrhovat a zapojovat.

Podstatou SSO je jakýsi program, který nám obvod bude řídit. Ten je dán právě kombinačním obvodem pro změnu vstupů. Co takový kombinační obvod dělá? Velmi stručně řečeno – paměťový obvod má vstupy a výstupy. **Kombinační obvod je sestaven tak, aby nastavil správné hodnoty vstupů paměťového obvodu podle hodnot výstupů paměťového obvodu.** Paměťový obvod si je poté zapíše na výstupy, když dostane hodinový pulz.

Kombinační obvod má **své vlastní vstupy**. Na ty jsou přivedeny **hodnoty z výstupů** paměťového obvodu. A **kombinační obvod** má také **výstupy**, které jsou přivedeny na **vstupy paměťového obvodu**. Lépe je to vidět na obrázku 1.5.



Obrázek 1.5 – Zapojení vstupů a výstupů mezi kombinačním a paměťovým obvodem

Kombinační obvod na obrázku musí nastavit své výstupy na správnou hodnotu podle hodnot svých vstupů. Podobný obvod už známe. Jsou to obvody logických funkcí. A právě proto budeme jako součásti pro sestavení kombinačního obvodu používat logické funkce. Připomenu, které známe: AND, NAND, NOT, OR, XOR.

Logická funkce mohla mít **více vstupů**, ale **výstup** byl vždy jen **jeden**. Hodnota výstupu se nastavovala podle **pravdivostní tabulky**. Pokud má mít kombinační obvod **více výstupů**, musí ho tvořit **více logických funkcí**. V našem počítačle využíváme dva vstupy paměťového obvodu, vstup A a vstup B. Kombinační obvod musí mít dva výstupy, musí ho tvořit dvě logické funkce. Ke každé funkci musíme vytvořit pravdivostní tabulku.

### Krok 3 – z grafu vyčteme pravdivostní tabulky

V našem obvodu počítačů potřebujeme dvě pravdivostní tabulky, protože tvoříme dvě logické funkce. Výstupy těchto logických funkcí připojíme na vstupy paměťového obvodu. Nazvu je logická funkce pro vstup A a logická funkce pro vstup B.

Po zapnutí má integrovaný obvod 74175 své výstupy A i B nastaveny na hodnotu log0. To je pro nás počáteční stav, značený 00 v binárním zápisu. Ten potřebujeme změnit na stav 01, čili A=0 a B=1. Dále měníme na stav 10, tedy A=1 a B=0, pak na stav 11, tedy A=1 a B=1. Nakonec chceme nastavit opět počáteční stav, tedy A=0 a B=0. Zapišu to do tabulky na obrázku 1.6.

HODNOTY VÝSTUPŮ 74175		NOVÉ HODNOTY PRO VSTUPY	
A	B	A	B
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

PŮVODNÍ HODNOTA VÝSTUPŮ 74175 →

POŽADOVANÁ HODNOTA PRO NÁSLEDUJÍCÍ STAV SSO ←

Obrázek 1.6 – Tabulka přechodů SSO

Vlevo je zapsán současný stav výstupů paměťového obvodu 74175. Vpravo je stav pro následující stav, podle grafu stavů z obrázku 1.1 nebo 1.2.

Aby bylo jasné, že sestavujeme dvě logické funkce, z tabulky přechodů vypíšu dvě pravdivostní tabulky. Jsou na obrázku 1.7.

VSTUPY LOGICKÉ FUNKCE		VÝSTUP LOGICKÉ FUNKCE
A	B	A
0	0	0
0	1	1
1	0	1
1	1	0

VSTUPY LOGICKÉ FUNKCE		VÝSTUP LOGICKÉ FUNKCE
A	B	B
0	0	1
0	1	0
1	0	1
1	1	0

Obrázek 1.7 – Pravdivostní tabulky hledaných logických funkcí

#### Krok 4 – Sestavit logické funkce podle pravdivostních tabulek

Známe několik funkcí, které mají i svou schematicou značku. A mají odpovídající pravdivostní tabulku. Logické funkce, které tvoříme zde, budou většinou úplně nové. Budeme je muset kombinovat ze známých logických funkcí (AND, OR, NOT, NAND).

V tomto případě - při návrhu počítačového obvodu - to nebude moc těžké. Logické funkce mají pravdivostní tabulky podobné některým, které už známe.

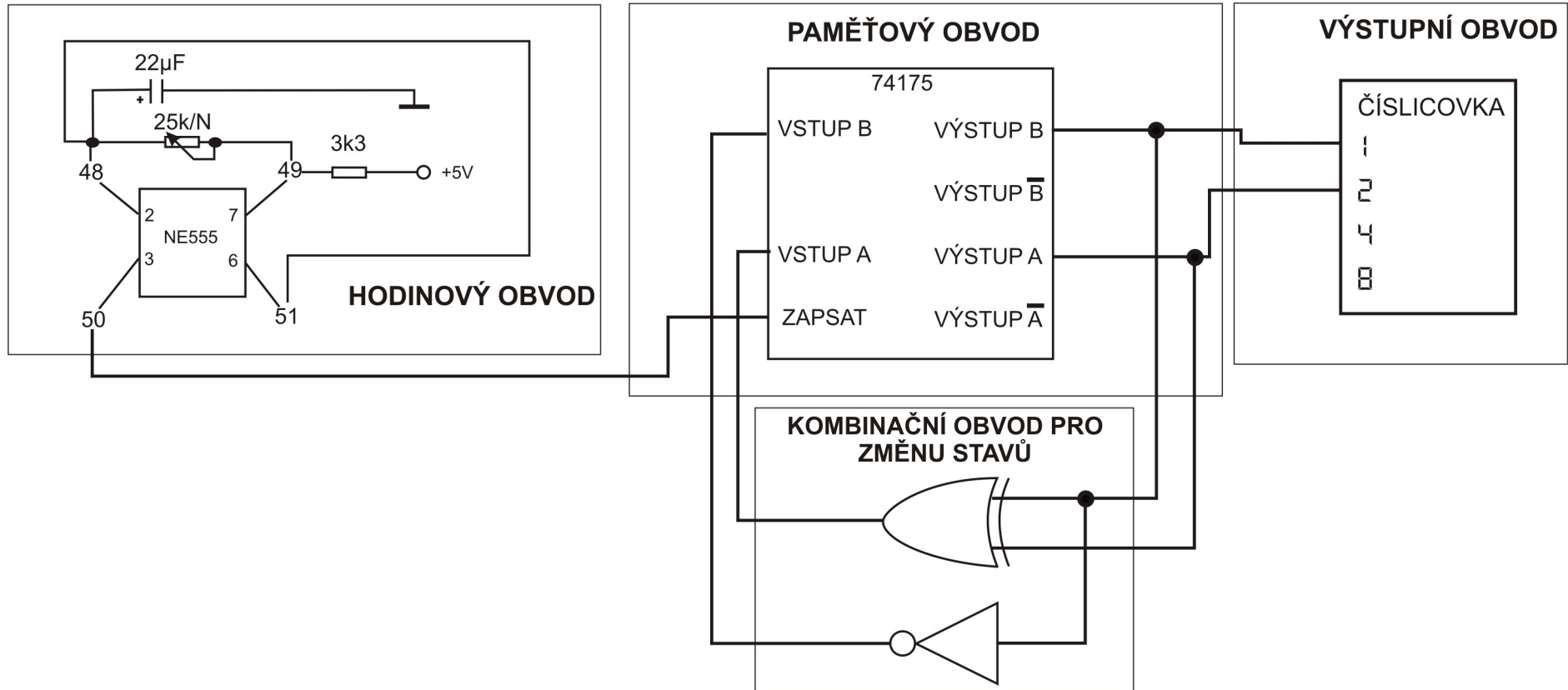
Tabulka pro vstup paměťového obvodu A (ta vlevo) odpovídá pravdivostní tabulce funkce XOR.

**Výstup logické funkce A = (vstup A XOR vstup B).**

Tabulka pro vstup paměťového obvodu B (vpravo) sice neodpovídá žádné známé funkci, ale když se pozorně podíváte, všimnete si, že výstup B je přesně opačný vstupu B. „Opačný“ - to je logická funkce NOT.

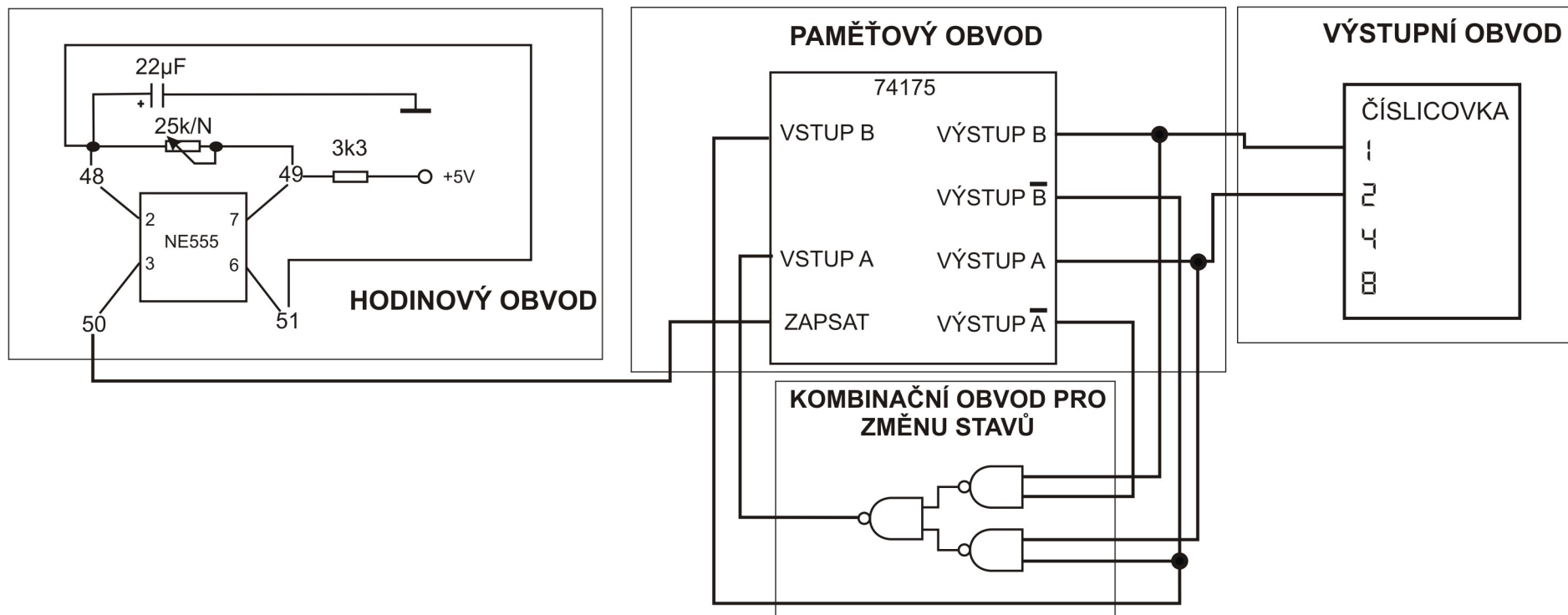
**Výstup logické funkce B = (NOT vstup B).**

Logické funkce pro kombinační obvod změny stavů jsou hotové. Teď stačí dokreslit schéma a obvod zapojit. Doplněné schéma je na obrázku 1.8.



Obrázek 1.8 – Kompletní blokové schéma počítadla

Funkci NOT B nemusíme zapojovat – integrovaný obvod 74175 má pro každý výstup i výstup negovaný. Na vstup B paměťového obvodu stačí připojit výstup B s čarou. Funkci XOR budeme muset sestavit z funkcí NAND. Návod je na obrázku 2.16 předchozího dílu návodu. Schéma počítadla, kde je kombinační obvod sestaven z funkcí NAND, je na obrázku 1.9.



Obrázek 1.9 – Blokové schéma počítadla s kombinačním obvodem sestaveným z funkcí NAND

**Úloha 1: Zapojení:**

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

*Paměťový obvod: 73-77, 75-76*

*Kombinační obvod pro změnu stavů: 74-62, 78-63, 64-71, 68-79, 69-75, 70-72*

*Výstupní obvod: 74-91, 79-90*

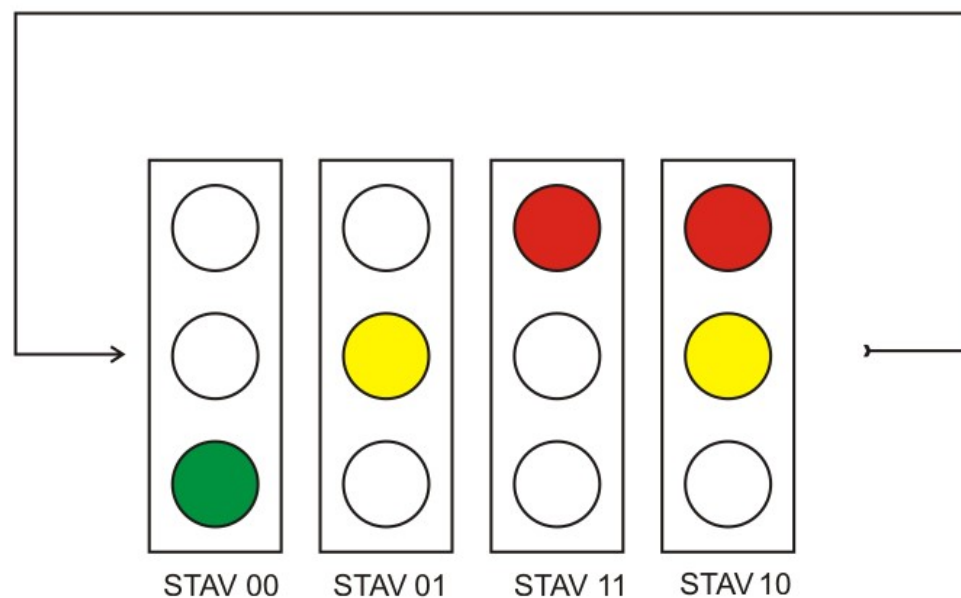
**Všechny IO připojte napájení na +5V. Toto už od této chvíle budu považovat za automatické.**

## Kapitola 2 – Semafor na přechodu pro chodce

Další SSO, který budeme navrhovat, jsem nazval „semafor“. Při návrhu budeme postupovat přesně podle kroků z předchozí kapitoly.

### Krok 1 – Slovní popis funkce semaforu

Semafor na vozovce pomocí barevných světel signalizuje několik situací. Popíšeme si každou zvlášť. Výchozí stav - na semaforu svítí zelená, auta mohou projíždět. Pokud chodec stiskne tlačítko pro chodce, semafor zhasne zelenou, rozsvítí oranžovou, ta chvíli svítí, pak zhasne oranžová a rozsvítí se červená. Auto musí zastavit a chodec může přejít vozovku. Dále se k rozsvícené červené přidá oranžová, auto se připravuje na jízdu, pak červená s oranžovou zhasnou a rozsvítí se opět zelená. A semafor je opět ve výchozím stavu, auta mohou projíždět. Obrázek 2.1 ukazuje, jak se světla na semaforu střídají.

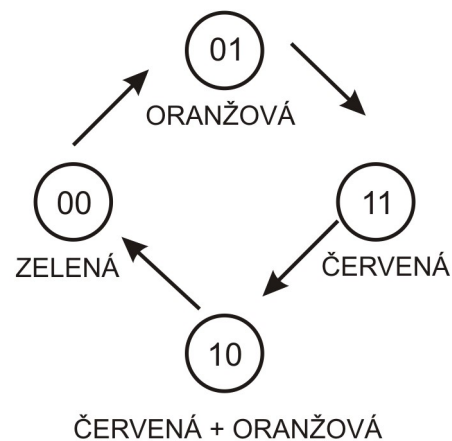


Obrázek 2.1 – Kombinace signalizace světel semaforu

Signalizace semaforu prochází čtyřmi stavy. Očíslování těchto stavů jsem k obrázku napsal ve dvojkové soustavě: 0=00bin, 1=01bin, 2=10bin, 3=11bin. Stavy se stále opakují. Číslování stavů je následující - zelená = stav obvodu 0. Oranžová = stav obvodu 1. Logické by teď bylo, kdyby následující stav, tedy rozsvícená červená, byl stav obvodu 2. Na obrázku ale vidíme, že číslo stavu je 11 binárně, tedy trojka. Je to jen proto, aby byl kombinační obvod pro změnu stavů jednodušší. **Pokud stavy, které jdou po sobě, se v binárním zápisu liší vždy jen změnou jednoho bitu, jsou logické funkce kombinačního obvodu méně složité.** Poznámka – stav 01 a stav 10 se liší v obou bitech, musíme změnit A i B. Všimněte si, že na obrázku se vždy následující stav od předchozího liší pouze změnou jedné jedničky nebo nuly na její opak. Samozřejmě by to šlo i číslováním stavů 0,1,2,3, ale takhle to bude jednodušší obvod.

### Krok 2 – nakreslíme si, jak jdou stavy po sobě

Číslo stavů nakreslíme do grafu a šipkami vyznačíme, jak jdou po sobě. Graf ukazuje obrázek 2.2. Poznámka – zatím vytváříme semafor bez tlačítka pro chodce. Takže obvod bude stavy měnit stále dokolečka.



Obrázek 2.2 – graf přechodů SSO semafor

Takovýto graf se dá nazvat konečný automat. Je to jakýsi předpis, který nám říká, jak se stavy střídají mezi sebou. Není to přesná definice konečného automatu. Účelem je pouze vědět, že něco takového existuje. Pro zájemce o lepší popis odkážu na webové stránky

[http://fel.jahho.cz/3.semestr/lob/prednasky/s\\_fsm1.pdf](http://fel.jahho.cz/3.semestr/lob/prednasky/s_fsm1.pdf)

[http://fel.jahho.cz/3.semestr/lob/prednasky/s\\_fsm2.pdf](http://fel.jahho.cz/3.semestr/lob/prednasky/s_fsm2.pdf)

### Krok 3 – z grafu vyčteme pravdivostní tabulky

Opět vlevo do tabulky píšu stávající stav a vpravo nově požadované hodnoty, které nám bude vytvářet kombinační obvod pro změnu stavů.

HODNOTY VÝSTUPŮ 74175		NOVÉ HODNOTY PRO VSTUPY	
A	B	A	B
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

PŮVODNÍ HODNOTA VÝSTUPŮ 74175

POŽADOVANÁ HODNOTA PRO NÁSLEDUJÍCÍ STAV SSO

Obrázek 2.3 – Tabulka přechodů semaforu

Potřebujeme vytvořit hodnoty pro dva vstupy paměťového obvodu. Logická funkce má vždy jen jeden výstup. Logické funkce, které budou tvořit kombinační obvod pro změnu stavů, budou opět dvě. Jejich pravdivostní tabulky jsou na obrázku 2.4.

VSTUPY		VÝSTUP
A	B	A
0	0	0
0	1	1
1	1	1
1	0	0

PRVNÍ LOGICKÁ FUNKCE

VSTUPY		VÝSTUP
A	B	B
0	0	1
0	1	1
1	1	0
1	0	0

DRUHÁ LOGICKÁ FUNKCE

Obrázek 2.4 – pravdivostní tabulky funkcí

#### Krok 4 – Sestavit logické funkce podle pravdivostních tabulek

Logické funkce můžeme opět odhadnout. Všimněte si, že v tabulce vlevo sloupec pro výstup je stejný jako sloupec pro vstup B. Můžeme napsat:  
**Výstup logické funkce A = vstup B.**

Pohledem na tabulku vpravo můžete vidět, že sloupec pro výstup má opačné hodnoty, než sloupec pro vstup A. Můžeme napsat:  
**Výstup logické funkce B = NOT (vstup A).**

Tím ale návrh nekončí. SSO pro semafor má totiž ještě jeden blok navíc. A ten se nazývá kombinační obvod pro výstupní obvod (potřebujeme rozsvěcet ledky). Které ledky budou svítit v různých stavech a které ne, to musíme určit z čísla stavu, v kterém SSO je (tedy z hodnot výstupů paměťového obvodu). Ledky jsou tři. Každou bude ovládat jedna logická funkce. Kombinační obvod pro výstupy bude složen ze tří logických funkcí.

### Krok 5 – Napsat pravdivostní tabulky pro logické funkce kombinačního obvodu ovládající výstupní (zobrazovací) obvod

Napsat pravdivostní tabulky logických funkcí, které budou ovládat ledky, je jednoduché. Musíme si uvědomit, v kterém stavu ledka svítí (log1). Nesvítí = log0. Tak jednoduše napíšeme 1 na řádek, ve kterém je správné číslo stavu. Pravdivostní tabulky jsou na obrázku 2.5.

STAV SSO		LEDKA
A	B	ZELENÁ
0	0	1
0	1	0
1	1	0
1	0	0

STAV SSO		LEDKA
A	B	ŽLUTÁ
0	0	0
0	1	1
1	1	0
1	0	1

STAV SSO		LEDKA
A	B	ČERVENÁ
0	0	0
0	1	0
1	1	1
1	0	1

Obrázek 2.5 – Pravdivostní tabulky logických funkcí pro ledky

### Krok 6 – Navrhnout kombinační obvod pro výstupní obvod

Začneme logickou funkcí pro zelenou ledku. Pravdivostní tabulka připomíná dvě složené funkce. Buď se na ni můžeme dívat jako na pravdivostní tabulku logické funkce OR, která má na výstup připojenou ještě funkci NOT. Pravý sloupec je totiž přesně obráceně, než má funkce OR. Nebo se můžeme dívat jako na funkci AND, která má před každým vstupem zapojenou funkci NOT. Sloupce vstupů totiž vypadají obráceně, než má funkce AND. Já si vyberu tuto druhou variantu, máme totiž k dispozici negované výstupy paměťového obvodu.

**Výstup logické funkce pro zelenou ledku = NOT A AND NOT B**

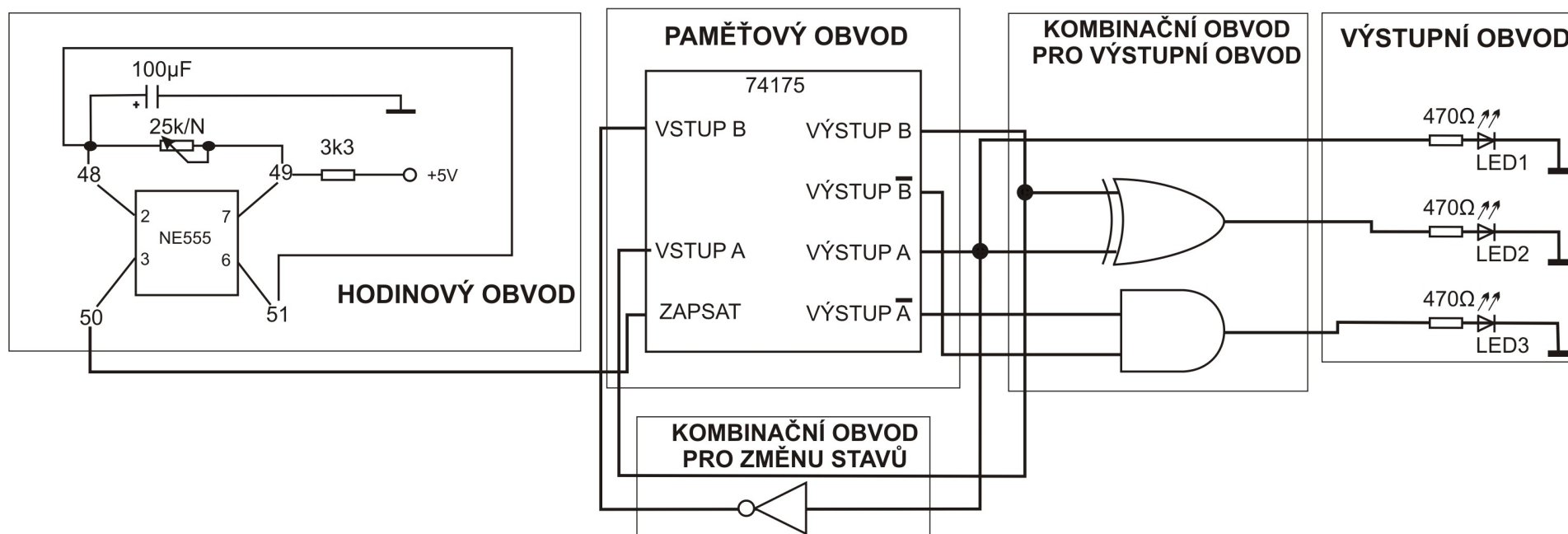
Logická funkce pro žlutou ledku – pravdivostní tabulku už známe. Funkce XOR.

**Výstup logické funkce pro žlutou ledku = A XOR B**

Červená ledka má stejné hodnoty jako výstup A.

**Výstup logické funkce pro červenou ledku = A**

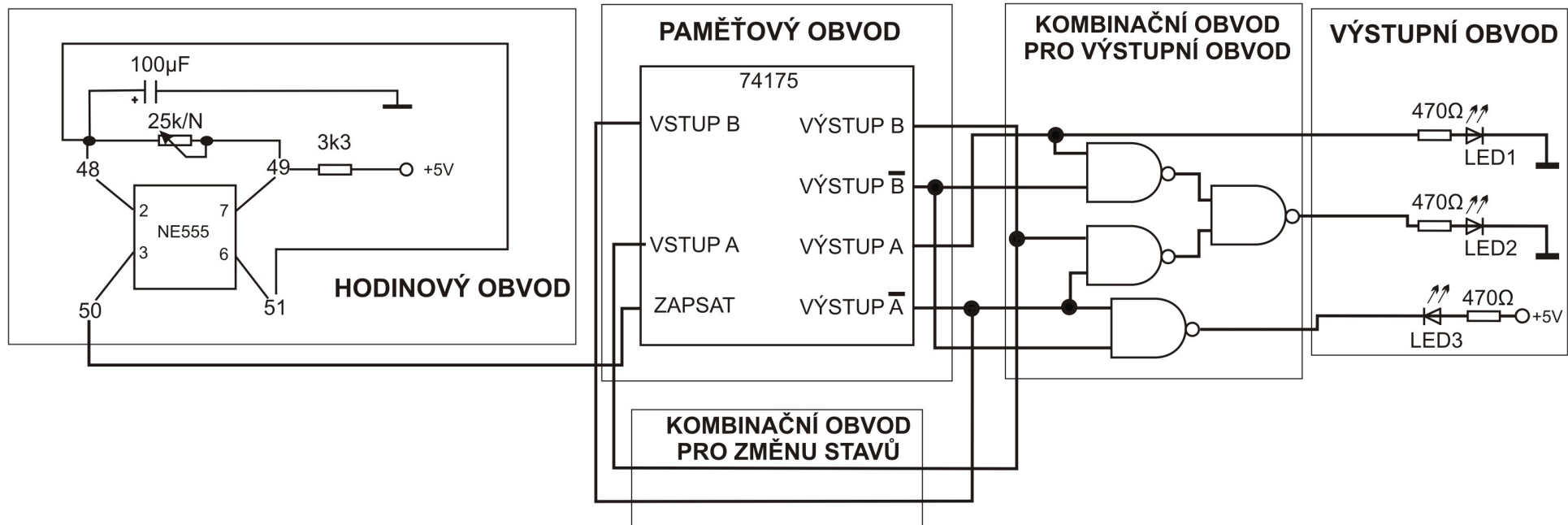
Ted' zbývá nakreslit blokové schéma obvodu, který jsme nazvali **semafor**.



Obrázek 2.6 – Blokové schéma SSO semafor

Vidíte, že kombinační obvod pro změnu stavů je o dost jednodušší, než v případě počítadla. Přitom se také mění jen čtyři stavy, jsou však výhodněji očíslované.

Na obrázku 2.7 je schéma, kde jsou logické funkce složeny z funkcí NAND, aby bylo možné semafor zapojit na stavebnici. Funkci AND pro zelenou ledku jsem musel šalamounsky upravit. Víme, že AND se dá složit pomocí dvou funkcí NAND. Ale protože budeme potřebovat tři NANDy pro složení XORu, tak nám pro AND zbývá jen jeden NAND. A dá se to zapojit – když si uvědomíme, že výstup AND je obráceně oproti NAND, tak pokud za NAND zapojíme ledku obráceně, tak se bude chovat, jako by byla zapojena na výstup funkce AND. XOR jsem složil tak, jak už to známe z předchozí úlohy.



Obrázek 2.7 – Blokové schéma SSO semafor s funkcemi NAND

### Úloha 2: Zapojení

Hodinový obvod: 49-10 , 9-log1, 48-51 , 23-48 , 24-49 , 48-41 , 40-log0 , 50-87

Kombinační obvod pro změnu stavů: 74-77, 76-78

Kombinační obvod pro výstupní obvod: 75-68, 69-79, 70-65, 66-73, 72-78, 71-74, 62-75, 63-78

Výstupní obvod: 97-log1, 92-94, 92-log0, 96-64, 95-67, 93-79,

Všechny IO připojte napájení na +5V.

Pokud chcete, připojte ještě 64-117, 116-80. Tím připojíte tlačítko pro chodce. Semafor bude stále ve stavu 00, a při stisku tlačítka proběhne celý cyklus. Když se semafor vrátí do stavu 00 – svítí zelená, tak pokud nebude tlačítko zmáčknuté, zůstane v tomto stavu.

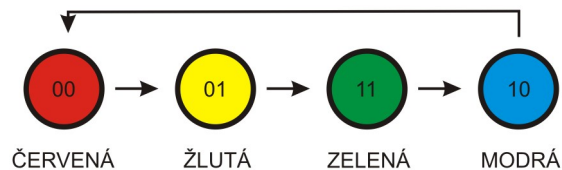
Tlačítko není přímo součástí návrhu SSO, spíše vychází z logické úvahy. IO 74175 totiž může měnit stavy, jen pokud vývod 80, značený SMAZAT, není připojen na log0. Využil jsem toho, že **pouze** ve stavu 00 je na zdírce 64 hodnota log0. A ta nám semafor „zastaví“.

## Kapitola 3 – Blikač „Poloviční Knight Rider“

### Krok 1 – Slovní popis funkce obvodu

Jako další a poslední úlohu jsem vymyslel blikač se čtyřmi ledkami, který rozsvítí vždy jen jednu ledku, ostatní budou zhasnuté. A bude rozsvěcet v pořadí červená, žlutá, zelená, modrá. Zkuste vyhledat na internetu, jak vypadá blikač Knight Rider (ze stejnojmenného seriálu 80tých let). Ledky blikají sem a tam. Náš obvod bude dělat jen polovinu, ledky budou blikat jen jedním směrem. K tomu nám vystačí 4 stavy. Očíslujeme je tak, jako v případě semaforu. Stav 00 bude výchozí stav, ve kterém svítí červená ledka. Stav 01 bude následovat, v tomto stavu bude svítit jen žlutá ledka. Další stav bude mít číslo 11 a bude svítit zelená ledka. Poslední stav bude mít číslo 10 a bude svítit modrá ledka. Pak bude následovat výchozí stav.

### Krok 2 – nakreslíme si, jak jdou stavy po sobě



Obrázek 3.1 – graf přechodů SSO „poloviční Knight Rider“

### Krok 3 – z grafu vyčteme tabulku přechodů a pravdivostní tabulky pro kombinační obvod změny stavů

Tohle máme vlastně hotové v předchozí úloze. Budou to stejné tabulky.

HODNOTY VÝSTUPŮ 74175		NOVÉ HODNOTY PRO VSTUPY	
A	B	A	B
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

Obrázek 3.2 – Tabulka přechodů SSO

VSTUPY LOGICKÉ FUNKCE		VÝSTUP LOGICKÉ FUNKCE
A	B	A
0	0	0
0	1	1
1	1	1
1	0	0

LOGICKÁ FUNKCE PRO VSTUP A

VSTUPY LOGICKÉ FUNKCE		VÝSTUP LOGICKÉ FUNKCE
A	B	B
0	0	1
0	1	1
1	1	0
1	0	0

LOGICKÁ FUNKCE PRO VSTUP B

Obrázek 3.3 – Pravdivostní tabulky kombinačního obvodu pro přechod stavů SSO

#### Krok 4 – Sestavit logické funkce podle pravdivostních tabulek

To je opět hotové z předchozí úlohy. Jen to opišu.

**Výstup logické funkce A = vstup B.**

**Výstup logické funkce B = NOT (vstup A).**

#### Krok 5 – Napsat pravdivostní tabulky pro logické funkce kombinačního obvodu ovládající výstupní obvod

VÝSTUPY 74175		ČERVENÁ LEDKA
A	B	VÝSTUP FUNKCE
0	0	1
0	1	0
1	1	0
1	0	0

VÝSTUPY 74175		ŽLUTÁ LEDKA
A	B	VÝSTUP FUNKCE
0	0	0
0	1	1
1	1	0
1	0	0

VÝSTUPY 74175		ZELENÁ LEDKA
A	B	VÝSTUP FUNKCE
0	0	0
0	1	0
1	1	1
1	0	0

VÝSTUPY 74175		MODRÁ LEDKA
A	B	VÝSTUP FUNKCE
0	0	0
0	1	0
1	1	0
1	0	1

Obrázek 3.4 – Pravdivostní tabulky kombinačního obvodu pro výstupní obvod SSO

Každá ledka svítí vždy jen v jednom stavu, podle jedničky v tabulce.

#### Krok 6 – Navrhnout kombinační obvod pro výstupní obvod

**Výstup logické funkce pro červenou ledku = (NOT A) AND (NOT B)**

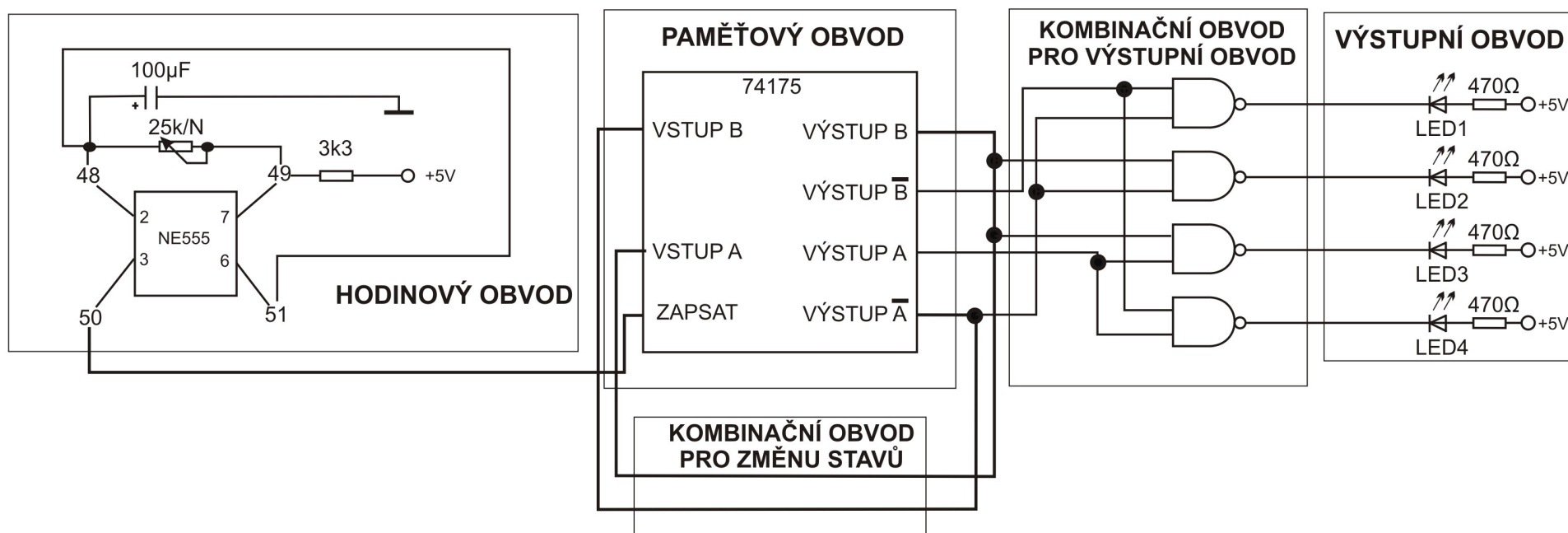
**Výstup logické funkce pro žlutou ledku = (NOT A) AND B**

**Výstup logické funkce pro zelenou ledku = A AND B**

**Výstup logické funkce pro modrou ledku = A AND (NOT B)**

Je jasné, jak jsem na funkce přišel? Podívejte se vždy na řádek s jedničkou ve sloupci „výstup funkce“. Tabulka pro zelenou ledku – musíme udělat logický součin negace A a negace B.

A ještě schéma zapojení. Je na obrázku 3.5. Jelikož nemáme funkce AND, použijeme stejný „trik“ s otočením ledky a funkcí NAND.



Obrázek 3.5 – blokové schéma SSO „poloviční Knight Rider“

### Úloha 3: Zapojení

*Hodinový obvod: 49-10 , 9-log1 , 48-51 , 23-48 , 24-49 , 48-41 , 40-log0 , 50-87*

*Kombinační obvod pro změnu stavů: 74-77 , 76-78*

*Kombinační obvod pro výstupní obvod: 78-71 , 74-63 , 68-74 , 78-62 , 65-79 , 79-69 , 75-72 , 75-66 , 64-94 , 67-98 , 73-92 , 70-96*

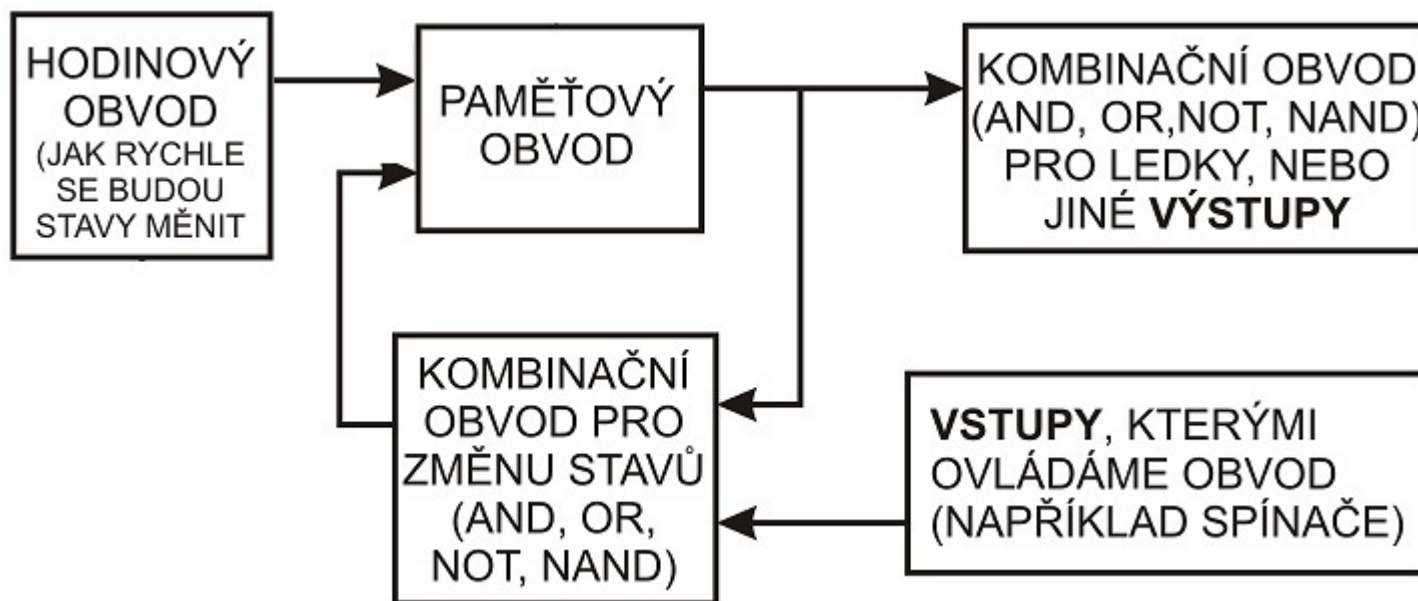
*Výstupní obvod: 93 - 95 , 95 - 97 , 97 - 99 , 99 -log1*

*Všechny IO připojte napájení na +5V.*

Když budete chtít měnit rychlost blikání, a nestačí regulace pomocí potenciometru, vyměňte kondenzátor 100µF za kondenzátor s jinou kapacitou. Můžete také připojit číslicovku pro sledování čísel stavů SSO. Připojte 79-90, 74-91, zapněte číslicovku.

## Kapitola 4 – Shrnutí tvorby SSO

Na závěr shrnutí – Hlavní myšlenkou stavebnice Saimon je návrh a zapojení synchronních sekvenčních obvodů, zkráceně SSO. Blokové schéma SSO je na obrázku 4.1. Je zde ještě navíc jeden poslední blok – **vstupy, kterými ovládáme obvod**. S tím budeme pracovat v příštím díle návodu, který se bude věnovat rozšiřujícímu modulu **Saimon 2 – Logické funkce**. Naučíme se pracovat s logickými funkcemi, které mají více vstupů, než dva, jako tomu bylo doposud. Tím budeme moci dosáhnout toho, že SSO bude mít více stavů, než jen čtyři – zatím jsme používali jen stavy 00, 01, 11 a 10. Tyto hodnoty jsme přiváděli na vstupy logických funkcí. A protože jsme měli k dispozici jen funkce se dvěma vstupy, byli jsme omezeni na čtyři stavy SSO. Naučíme se také, jak sestavit logické funkce pomocí Karnaughových map (čti Karnafových). Mapy budeme používat v případě, že nebude pohledem na pravdivostní tabulku zřejmé, jaká je to funkce.



Obrázek 4.1 – Teoretický popis SSO - shrnutí

## Obsah

Kapitola 1 - Obvod, který „žije“ – co si pod tím představit.....	2
Kapitola 2 – Semafor na přechodu pro chodce .....	11
Kapitola 3 – Blikač „poloviční Knight Rider“ .....	18
Kapitola 4 – Shrnutí tvorby SSO .....	22