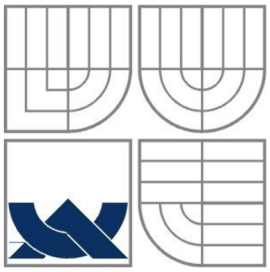
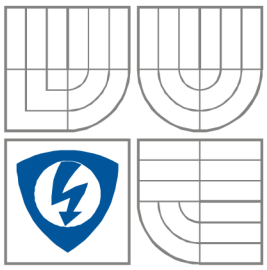


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

ČÍSLICOVÁ TECHNIKA V LABORATORNÍ VÝUCE

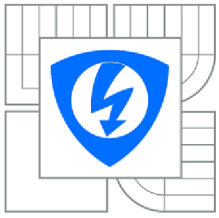
DIGITAL TECHNIQUE IN LABORATORY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Michal Harvánek
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Tomáš Frýza, Ph.D.
SUPERVISOR

BRNO, 2013



VYSOKÉ UCENÍ TECHNICKÉ V BRNE

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor

Elektronika a sdělovací technika

Student: Michal Harvánek ***ID:*** 133507
Ročník: 3 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Číslicová technika v laboratorní výuce

POKYNY PRO VYPRACOVÁNÍ:

Projekt je orientován na základy číslicové techniky. Prostudujte stávající laboratorní cvičení předmětu Impulzová a číslicová technika. Navrhnete témata a zapojení nových úloh zaměřených především na syntézu kombinačních obvodu, asynchronních a synchronních čítačů, příp. stavových automatu. Zapojení realizujte pomocí stavebnicového systému firmy RC Didactic. Vypracujte vzorové protokoly vámi navržených úloh.

Navrhnete a naprogramujte jednoduché Java Applety na vybraná témata číslicové techniky (např. čítače, optimalizace KLF pomocí map, převodníky, aj.). Orestujte správnou funkci vašich appletů, umístěných na internetu.

DOPORUCENÁ LITERATURA:

[1] FRÝZA, T. Stránky předmětu Impulzová a číslicová technika [online]. 2012 - [cit. 14. května 2012]. Dostupné na WWW: http://www.urel.feec.vutbr.cz/~fryza/.

[2] Stránky firmy RC Didactic [online]. 2012 - [cit. 14. května 2012]. Dostupné na WWW: http://www.rcdidactic.cz/cz/.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 31.5.2013

Vedoucí práce: doc. Ing. Tomáš Frýza, Ph.D.
Konzultanti semestrální práce:

prof. Dr. Ing. Zbyněk Raida

Předseda oborové rady

UPOZORNENÍ:

ABSTRAKT

Výsledkem bakalářské práce jsou návrhy kompletních laboratorních úloh pro demonstraci teoretických poznatků z oblastí synchronních, asynchronních čítačů, stavových automatů, speciálních kombinačních obvodů a hazardů vyskytujících se v kombinačně logických funkcích. V bakalářské práci je využit výukový systém od firmy RC Didactic, jehož hlavní výhodou je názorná ukázka funkce logických obvodů a jednoduchá počítačová aplikace, využitelná při analýze digitálních i analogových obvodů. Úlohy jsou navrženy tak, aby je bylo možno zapojit v dvouhodinovém časovém limitu laboratorních cvičení předmětu BICT. Součástí vypracování úloh jsou jednoduché java applety vypracované v prostředí NetBeans IDE, jež jsou umístěny na webových stránkách. Applety simulují funkce některých digitálních obvodů případně minimalizace Karnaughových map. Jednotlivé digitální obvody jsou dále simulovány v prostředí Xilinx ISE Design Suite.

KLÍČOVÁ SLOVA

Synchronní, asynchronní čítače, stavové automaty, speciální kombinačně obvody, hazardy, RC 2000, java applety, Quine-McCluskeyho algoritmus, NetBeans, Xilinx ISE Design Suite.

ABSTRACT

The result of this bachelor's thesis are complete proposals labs to demonstrate theoretical knowledge in the field of synchronous and asynchronous counters, state machines, combinational circuits and special hazards occurring in combinational logic functions. The bachelor's thesis utilized educational system from RC Didactic, whose main advantage is the illustrative proof of the logic circuits and a simple computer application utilized in the analysis of digital and analog circuits. Designe tasks can be involved in a two-hour time limit labs of BICT course. Tasks include simple java applets developed in NetBeans IDE. Tasks are placed on the website. Applets simulate the function of certain digital circuits or minimization Karnaugh maps. Individual digital circuits are also simulated in the Xilinx ISE Design Suite.

KEYWORDS

Synchronous, asynchronous counters, state machines, special combinational circuits, hazards, RC 2000,java applets, Quine-McCluskey algorithm, NetBeans, Xilinx ISE Design Suite.

Harvánek, M. *Číslicová technika v laboratorní výuce*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2013. 26 s., 37 s. příloh. Bakalářská práce. Vedoucí práce: doc. Ing. Tomáš Frýza, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svoji bakalářskou práci na téma Číslicová technika v laboratorní výuce jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a~jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce doc. Ing. Tomáš Frýza, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce. Dále děkuji RNDr. Jakub Valčík za odbornou pomoc a cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Cílem této bakalářské práce je navrhnout sadu laboratorních úloh pro kurz BICT – impulzová a číslicová technika, probíhajícím v letním semestru druhého ročníku bakalářského studijního programu. Úlohy jsou navrženy tak, aby je studenti byli schopni realizovat během dvouhodinového časového limitu. K realizaci úloh je využit výukový systém od firmy RC Didadctic, který umožňuje názorné a přehledné zapojení jednotlivých úloh.

První kapitola je zaměřena na problematiku hazardů u kombinačně logických funkcí. Jednotlivé úkoly laboratorní úlohy jsou navrženy tak, aby se student prakticky seznámil s hazardy u elementárních struktur, dynamickými a statickými hazardy. Protože jsou hazardy rychlé jevy, je u laboratorní úlohy zabývající se hazardy, použit externí osciloskop. Důvodem je malá vzorkovací frekvence osciloskopu, který je součástí stavebnice RC 2000.

Ve druhé kapitole a tomu odpovídající druhé laboratorní úloze se studenti seznámí s realizací kombinačních logických funkcí pomocí speciálních obvodů a to multiplexoru a obvodů typu NAND. Úkolem odpovídající laboratorní úlohy je návrh dvou bitové sčítačky a ověření její funkce na logickém analyzátoru. Studenti zapojí i integrované zapojení dvoubitové sčítačky.

Třetí kapitola se věnuje syntéze asynchronních, synchronních čítačů a stavových automatů. V příslušné laboratorní úloze se studenti seznámí s návrhem asynchronního tříbitového čítače realizovaného pomocí obvodu typu JK, synchronního integrovaného čítače a navrhnou 3 bitový stavový automat pomocí klopného obvodu typu D.

Dále jsou součástí této práce java applety umístěné na internetu, které popisují funkce daných číslicových obvodů. A applet Karnaughovy mapy, který je schopen minimalizovat logické funkce svou, tří a čtyř vstupných proměnných do obou způsobů zápisů výstupní funkce.

Jednotlivé digitální obvody, které jsou vytvořeny jako java applety, jsou dále simulovány v prostředí ISE Xilinx Design Suite. Zdrojový kód, popisující funkce jednotlivých digitálních obvodů, je psán formou HDL.

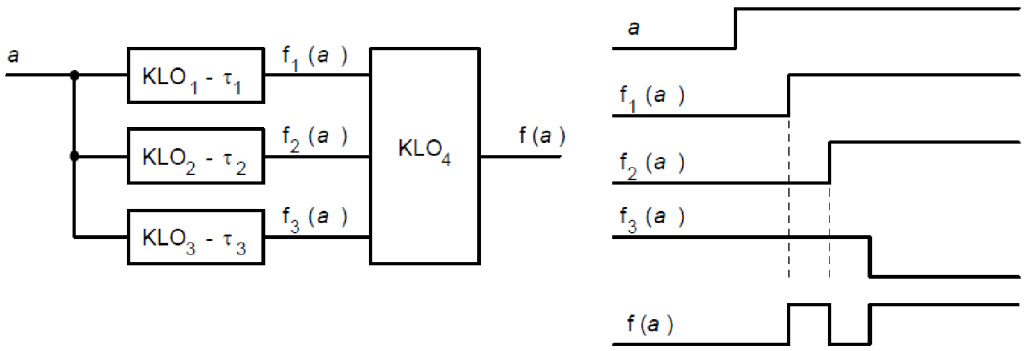
1 HAZARDY

Hazardy jsou stavy v kombinačně logických obvodech, při kterých v důsledku zpoždění jednotlivých obvodů může (nemusí) dojít ke vzniku parazitních impulzů na výstupu tohoto obvodu. Doba trvání těchto impulzů je především závislá na teplotě, velikosti napájecího napětí a na podobných vlivech, proto přesné zpoždění jednotlivých logických obvodů neznáme. V katalozích se pak udává jen hraniční hodnota. Parazitní impulzy, které se mohou objevit na výstupu kombinačně logických obvodů, označíme jako GLITCH. Hazardy můžeme rozdělit na dva základní typy a to hazardy dynamické a hazardy statické.

Statické hazardy se projevují u signálů, u nichž očekáváme stálou logickou úroveň, ale při změně sledované vstupní veličiny může na výstupu kombinačně logické funkce vzniknout parazitní impulz opačné polarity.

1.1 Dynamický hazard

Dynamické hazardy se projevují tehdy, očekáváme-li při změně vstupní veličiny změnu veličiny výstupní. Zpravidla očekáváme, že jedna změna vstupní veličiny způsobí jednu změnu výstupní veličiny. Je-li však v obvodu dynamický hazard, může se odezva výstupní veličiny skládat, to znamená, že při změně z jedné logické úrovně do druhé se mohou před ustálením logické hodnoty objevit parazitní impulzy (viz Obr. 1.1), které mohou ovlivnit výslednou logickou funkci obvodu.

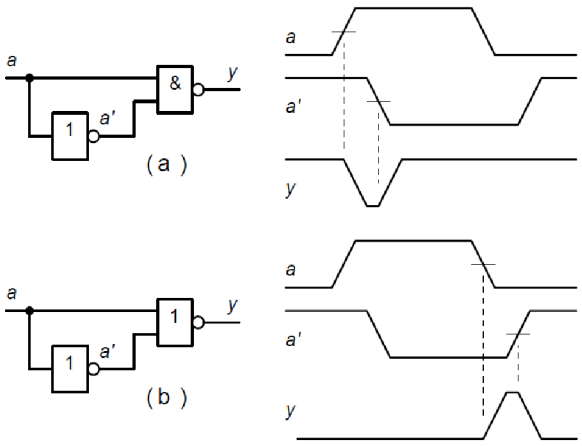


Obr. 1.1: Vznik parazitního impulsu při dynamickém hazardu (převzato z [1]).

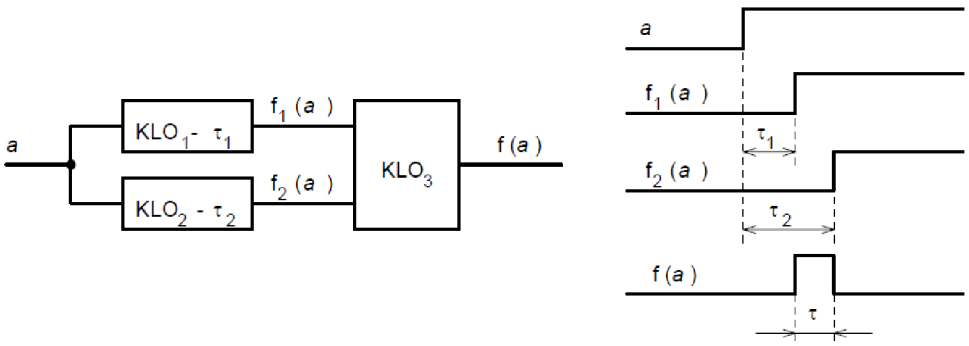
1.2 Statický hazard

Hazardy jsou nedílnou součástí návrhu kombinačních logických funkcí a pro jejich analýzu používáme dvou základních přístupů a to přístupu algebraického a přístupu za pomocí použití Karnaughových map. Hazardy u dvoustupňových struktur mohou nastat tehdy, obsahuje-li daná struktura takzvané elementární struktury s hazardem. Tyto struktury mohou být vícevstupové. Pokud v obvodu posuzujeme hazardy při změně jedné proměnné, můžeme tyto vícevstupové obvody nahradit

jednoduššími. V zapojení s členy NAND vzniká při hazardu parazitní impuls úrovně L při klidové úrovni H (viz Obr. 1.2a)., u zapojení se členy NOR je tomu naopak (viz Obr. 1.2b). U dvoustupňových struktur mluvíme o statickém hazardu, jehož příčinou jsou rozdílné zpoždění obvodů v paralelních větvích vstupujících do klopného obvodu. Toto zapojení je nakresleno na Obr. 1.3.



Obr. 1.2: Elementární struktury statického hazardu (převzato z [1]).



Obr. 1.3: Vznik parazitního impulsu při statickém hazardu (převzato z [1]).

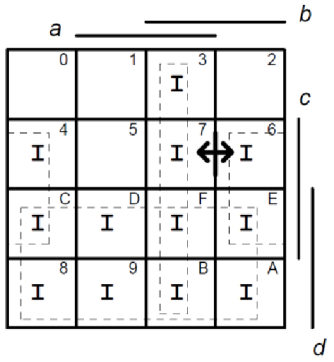
1.3 Potlačení hazardů

Pro správnou funkci obvodu je nezbytné, aby tyto stavy byly co nejvíce potlačeny. K tomuto můžeme přistoupit několika způsob:

1. Zpožděním výstupního signálu a jeho použití, až po uplynutí určité doby, po které se výstup ustálí.
2. Použití synchronních obvodů, ve kterých se parazitní impulzy nevyskytují.
3. Je možno použít filtru RC s charakterem integračního článku, ale tento postup rozhodně nelze doporučit jako vhodný pro globální řešení hazardů v kombinačně logických obvodech. Tento postup se používá jen zřídka a v nouzi.

Při návrhu kombinačně logické funkce pomocí hradel lze hazard odstranit přidáním termu do výsledné logické funkce. Tohoto můžeme dosáhnout různými způsoby. Prvním způsobem je řešení hazardů pomocí Karnaughových map. Druhým způsobem řešení odstranění hazardů je algebraický přístup.

Při odstraňování hazardů pomocí Karnaughových map vycházíme z faktu, že hazard může nastávat v případě, že v mapě sousedí dvě smyčky. Odstranění těchto hazardů provedeme tak, že místo hazardu – styk sousedních smyček, obepneme další smyčkou. Touto smyčkou přidáme do logické funkce další konsensu, který hazard potlačí. Názorná ukázka vyznačeného hazardu v Karnaughově mapě je na Obr. 1.4. Při realizaci kombinačně logické funkce pomocí obvodu NOR postupujeme stejným způsobem, jen s rozdílem, že namísto realizace smyček obsahující jedničky, realizujeme smyčky obsahující nuly.



Obr. 1.4: Mapa funkce s vyznačením hazardu při y = I (převzato z [1]).

2 SYNTÉZA KOMBINAČNÍCH LOGICKÝCH FUNKCÍ POMOCÍ SPECIÁLNÍCH OBVODŮ

Realizací kombinačně logické funkce rozumíme sestavení schématu zapojení pomocí příslušných obvodů, které ze vstupních proměnných vytvoří výstupní proměnnou s požadovanou logickou funkcí. V praxi se pro realizaci kombinačně logické funkce často používá pouze jeden integrovaný obvod. Tento se vyrábí sériově a najdeme ho v katalogu nebo se sériově nevyrábí a k realizaci funkce využijeme paměti (PROM) nebo obvody PLD. Přesto se v základu vždy vychází ze zápisu logické funkce pomocí tvaru SOP (Sum Of Products) nebo POS (Product Of Sums).

Nejčastější způsoby realizace kombinačně logických funkcí v číslicové technice jsou:

- pomoci integrovaných obvodů typu NAND, INVERT, NOR
- pomocí multiplexorů a demultiplexorů
- pomocí speciálních obvodů – převodníky kódu, násobičky, sčítačky
- pomocí pamětí PROM nebo programovatelných obvodů typu PLD

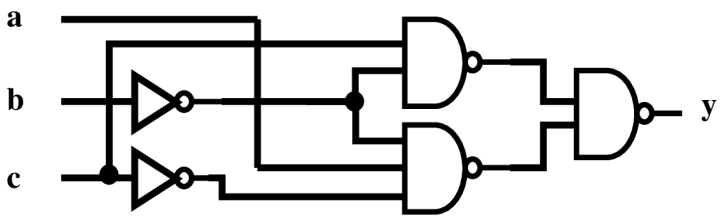
V této kapitole využijeme k realizaci obvodů především obvodů typu NAND (součinový tvar funkce) a multiplexorů. Výhodou zapojení pomocí obvodů typu NAND je, že pokud nevyužijeme všechny členy NAND v jednom pouzdře při zapojení první logické funkce, můžeme tyto obvody využít i při zapojení následující logické funkce.

2.1 Realizace pomocí obvodů typu NAND

Pro realizaci obvodů pomocí této metody vycházíme z Karnaughovy mapy do které zapíšeme hodnoty výstupní funkce. Poté tuto funkci zapíšeme v minimalizovaném tvaru (SOP). Na tuto funkci aplikujeme DeMorganovo pravidlo, pomocí kterého převedeme tvar s logickými součty na tvar s logickými součiny:

$$y = \overline{b} \cdot c + a \cdot \overline{b} \cdot \overline{c} \Rightarrow DeMorganovo\ pravidlo \Rightarrow \overline{\overline{\overline{bc}} \cdot \overline{a\overline{b}\overline{c}}}$$

Výslednou logickou funkci můžeme po této úpravě zapojit pouze pomocí obvodů typu NAND (viz Obr. 2.1).



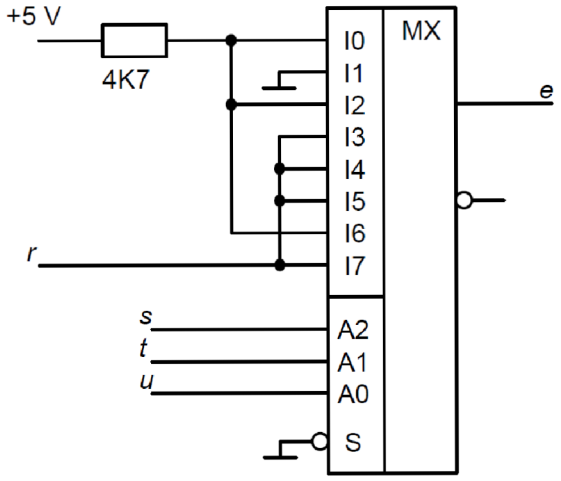
Obr. 2.1: Zapojení kombinačně logické y pomocí obvodů typu NAND.

2.2 Realizace pomocí multiplexoru

U realizace obvodů pomocí tohoto způsobu je postup poněkud komplikovanější. Podrobný postup je uveden ve skriptech[1] na straně 91. U realizace pomocí této metody se vyhází ze základu funkce multiplexoru. Každý multiplexor obsahuje adresové, datové vstupy a vstup S – select, který povoluje, popřípadě zakazuje přepis hodnoty na datovém vstupu na výstup. Tohoto se využívá při skládání multiplexorů do větších funkčních celků. Počet datových vstupů závisí na počtu adresových vstupů: počet datových vstupů = 2^N, kde N je počet adresových vstupů. Adresové vstupy adresují příslušný datový vstup, který má být převeden na výstup. Tohoto se právě využívá u realizace kombinační logické funkce, kdy na datové vstupy připojíme v závislosti na příslušné funkce buď +5V nebo GND. Tímto zajistíme, že při příslušných vstupních proměnných, které jsou přivedeny na adresové vstupy, dostaneme na výstupu požadovanou logickou hodnotu.

Mohlo by se na první pohled zdát, že u multiplexoru s N adresovými vstupy můžeme vytvořit obvod pouze pro N vstupních proměnných, ale v praxi se využívá zapojení, ve kterých realizujeme pomocí multiplexoru s N adresovými vstupy logickou funkci, která obsahuje N+1 vstupních proměnných. Postup návrhu takovýmto způsobem realizujeme následovně.

V prvním kroku výraz pro výslednou funkci doplníme tak, aby každý sčítanec obsahoval všechny eliminované proměnné. To znamená, že každý součinový člen vynásobíme závorkou, která obsahuje přímou i negovanou hodnotu chybějící eliminované proměnné. Ve druhém kroku závorky roznásobíme a vytkneme společné eliminované funkce. Ve třetím kroku všechny tyto funkce doplníme do tabulky, jejíž vstupní hodnoty budou eliminované funkce a výstupní hodnoty budou určeny zbytkovou funkcí. Zapojení kombinačně logické funkce pomocí multiplexoru je uvedeno na Obr. 2.2.



Obr. 2.2: Příklad zapojení kombinačně logické funkce pomocí multiplexoru (převzato z [1]).

3 ASYNCHRONNÍ, SYNCHRONNÍ ČÍTAČE A STAVOVÉ AUTOMATY

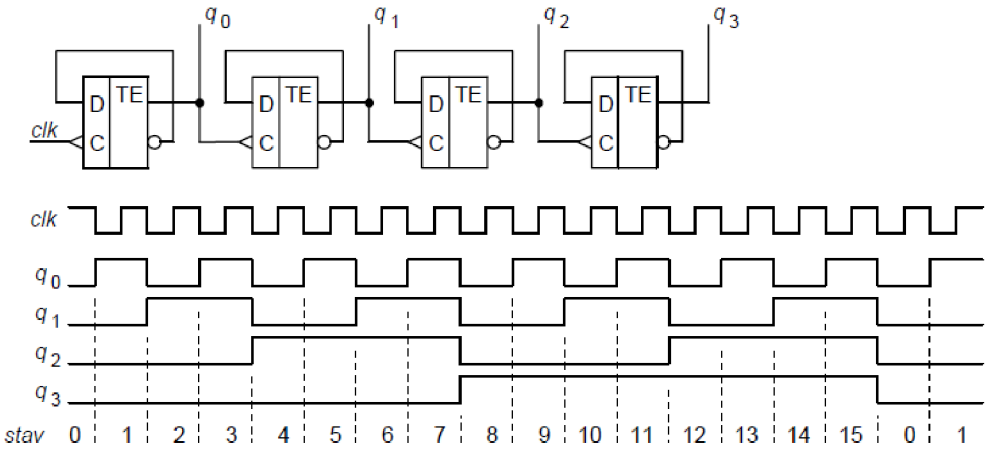
Čítače tvoří velkou podskupinu číslicových sekvenčních systémů. Čítače můžeme rozdělit dle různých hledisek, z nichž nejčastější jsou:

- Podle kódu v němž pracují – BCD, Gray, binární, dekadické…
- Podle způsobu zapojení – synchronní, asynchronní
- Podle směru čítaní – vpřed, vzad
- Podle typu použitých obvodů v části registrů

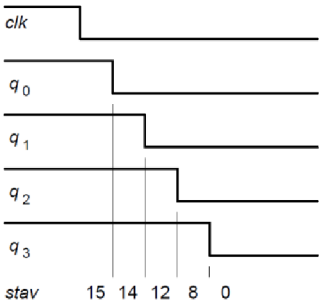
Nejčastěji se setkáváme s čítači binárními a dekadickým. Tyto čítače se vyrábějí v nejrůznějších integrovaných variantách vybavené příslušnými řídícími vstupy a výstupy. Mezi nejznámější vstupy/výstupy patří SET - nastavení, RESET - nulování, CARRY - přetečení, BORROW - podtečení, ENABLE - povolení. Z hlediska konstrukce a vnitřního zapojení se budeme zajímat především o rozdělení na synchronní a asynchronní čítače.

3.1 Asynchronní čítače

Asynchronní čítače se vyznačují tím, že hodinové vstupy jednotlivých klopných obvodů jsou zapojeny z výstupu předchozích klopných obvodů. Díky zpoždění jednotlivých klopných obvodů se může celkové zpoždění čítače nabývat větších hodnot než doba jedné periody vstupního kmitočtu. Jednotlivé klopné obvody se ustalují postupně => tyto čítače označujeme jako ripple counter (čítače s postupnou vlnou viz Obr 3.2). Jako základní zapojení můžeme uvažovat například čtyřbitový binární čítač vpřed z obvodu typu D. Z pravdivostní tabulky pro čítání vpřed je zřejmé, že změna vyššího bitu je podmíněná změnou bitu z 1 do 0 bitem nižším. Z tohoto vyplívá, že vstup CLK následujícího klopného je propojen s negovaným výstupem předchozího klopného obvodu. Na první pohled se může zdát, že asynchronní čítače jsou jednoduché na zapojení (viz Obr 3.1), ve skutečnosti již malá změna v čítacím pořadí popřípadě zkrácení cyklu vyžaduje složitější uvažování a analýzu obvod. Pro zapojení čítače z klopných obvodů JK se postupuje pomocí tabulek pro klopný obvod JK. Podrobný popis naleznete ve skriptech[1].



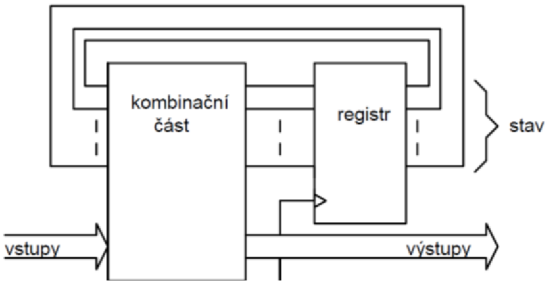
Obr. 3.1: Zapojení čtyřbitového asynchronního čítače vpřed pomocí klopných obvodů typu D (převzato z [1]).



Obr. 3.2: Postupný přechod asynchronního čítače ze stavu 15 do 0 (převzato z [1]).

3.2 Synchronní čítače a stavové automaty

Synchronní čítače a stavové automaty se vyznačují především tím, že jejich hodinové vstupy jsou zapojeny paralelně na řídící hodinový signál. Dále tyto obvody obsahují kombinační část a registr. Kombinační část v podstatě určuje posloupnost stavů na výstupu (viz Obr. 3.3), proto můžeme čítače i stavové automaty zahrnout do jedné skupiny. Jelikož jsou všechny registry řízeny ve stejném okamžiku nástupnou nebo sestupnou hranou hodinového signálu, případné hazardní stavy v kombinační části mají mezi dvěma následujícími sestupnými hranami dostatek času, aby se ustálily. Návrh synchronního čítače nebo stavového automatu se dá jednoduše algoritmizovat. Pravdivostní tabulka návrhu se skládá v podstatě ze tří částí (viz Tab. 3.1). V první části tabulky vypíšeme postupně posloupnost čítání. Ve druhé části vypíšeme do tabulky následující stavy čítání vzhledem k první části tabulky. Ve třetí části zapíšeme budící funkci. U obvodu typu D je budící funkce rovna následujícímu stavu (viz Tab. 3.2), takže se výsledná tabulka skládá v podstatě ze dvou částí. Charakteristická rovnice obvodu typu D je potom: $Q_{N+1} = D$. U obvodu typu JK vždy vycházíme z tabulky změn stavů výstupu obvodu v závislosti na hodnotách J a K (viz Tab. 3.3) tomu odpovídá i charakteristická rovnice obvodu JK: $Q_{N+1} = J\overline{Q} + KQ$.



Obr. 3.3: Obecné blokové schéma synchronního systému (převzato z [1]).

Tab. 3.1: Vzor pravdivostní tabulky návrhu synchronního systému.

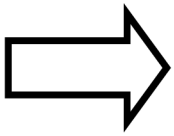
	Předchozí stav			Následující stav			Kombinační část
č.	**b2**	**b1**	**b0**	**b2**	**b1**	**b0**	**Doplnění podle typu klopného obvodu JK - tabulka zpětných hodnot viz Tab. 3.3 D - tabulka shodná s následujícím stavem viz Tab. 3.2**
1	0	0	0	**Doplnění podle požadované logické funkce**			
2	0	0	1				
3	0	1	0				
4	0	1	1				
5	0	0	0				
6	0	0	1				
7	0	1	0				
8	0	1	1				
9	1	0	0				
10	1	0	1				
11	1	1	0				
12	1	1	1				
13	1	0	0				
14	1	0	1				
15	1	1	0				
16	1	1	1				

Tab. 3.2: Odvození zpětné funkční tabulky pro obvod D.

Q	Q_{N+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Tab. 3.3: Odvození zpětné funkční tabulky pro obvod JK.

J	K	Q	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Q	Q_{N+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

4 VÝUKOVÝ SYSTÉM RC 2000 - µLAB

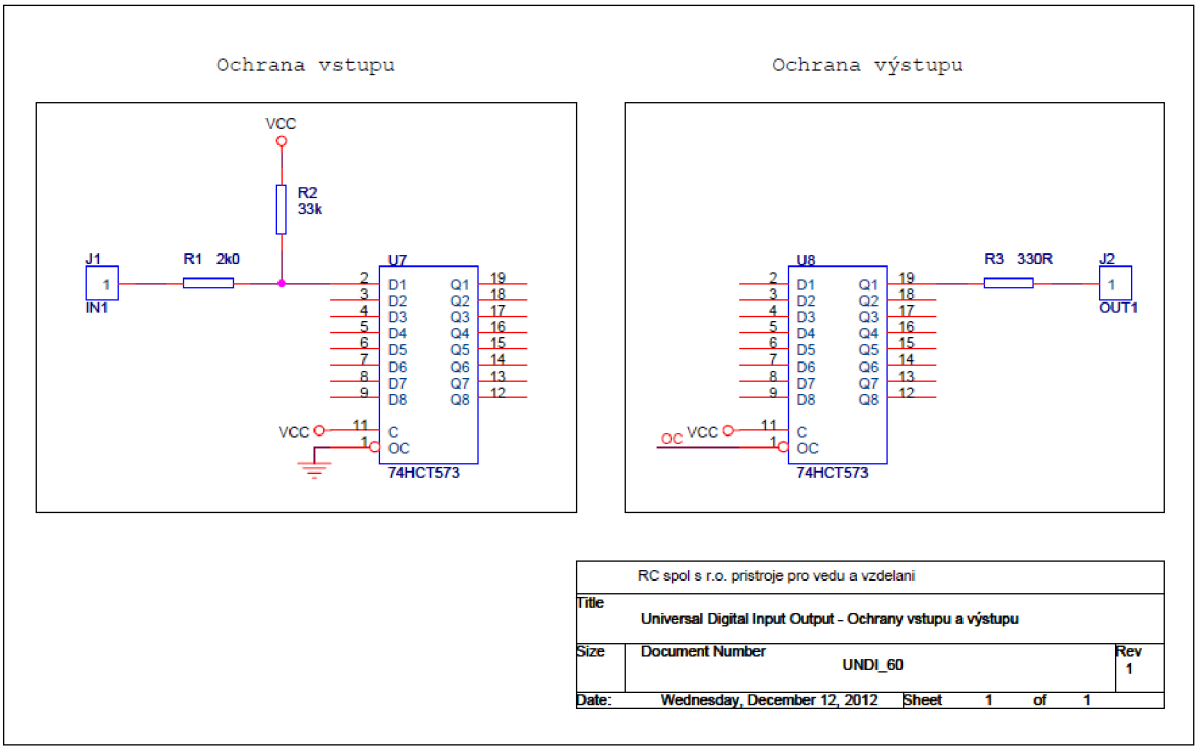
Výuka na systému RC 2000 je založena na reálném experimentu s podporou počítače. Moderní technologie, ochrana a přesnost jednotlivých modulů systému vede k souladu teoretické výuky s výsledky experimentu, tj. měření jsou „ideální“. Definovanou změnou obvodu je možno realizovat situaci, která by nastala při použití méně přesných součástek. „Reálný“ experiment pěstuje ve studentech cit pro elektroniku a vede ke schopnosti lépe využívat získané poznatky při další práci. Prioritou je důraz na vysvětlení základních principů elektrotechniky a elektroniky.

Výukový systém věnuje velkou pozornost didaktickým vlastnostem, zejména možnosti různých měřicích a zobrazovacích způsobů a jejich vzájemného porovnávání.

Sestavování měřicích zapojení je názorné, ovládání mikrolaboratoře intuitivní a měření je prezentováno přehledným způsobem. Systém šetří čas a umožňuje tak hlubší studium problémů.

Tento systém umožňuje experimenty v oblasti číslicové techniky, analogové techniky i v oblasti řízení a regulace. Více o tomto systému naleznete na stránkách výrobce http://www.rcdidactic.cz/cz/.

Výhodou systému RC 2000 je ochrana vstupů/výstupů, takže při chybném zapojení univerzálního číslicového modulu nedojde k jeho destrukci. Zapojení ochrany vstupů/výstupů je uvedeno na Obr. 4.1.



Obr. 4.1: Schéma zapojení ochrany vstupů/výstupů univerzálního číslicového modulu (převzato z [4]).

5 PROGRAMOVÁNÍ V JAVA

Cílem bakalářské práce je taktéž zhotovení jednoduchých java appletů, které simulují funkce daných digitálních obvodů a appletu, jež dokáže minimalizovat logické funkce dvou, tří nebo čtyř vstupních proměnných a to jak do tvaru SOP – sum of products, tak do tvaru POS – product of sums. Applety digitálních obvodů dále obsahují okno, ve kterém je příslušná funkce digitálního obvodu realizovaná pomocí jazyku HDL.

Java applety jsou naprogramovány v prostředí NetBeans IDE [8], což je úspěšný Open Source projekt s velmi rozsáhlou uživatelskou základnou a rostoucí komunitou vývojářů. Toto prostředí dále poskytuje příjemné uživatelské prostředí a vynikající debugger.

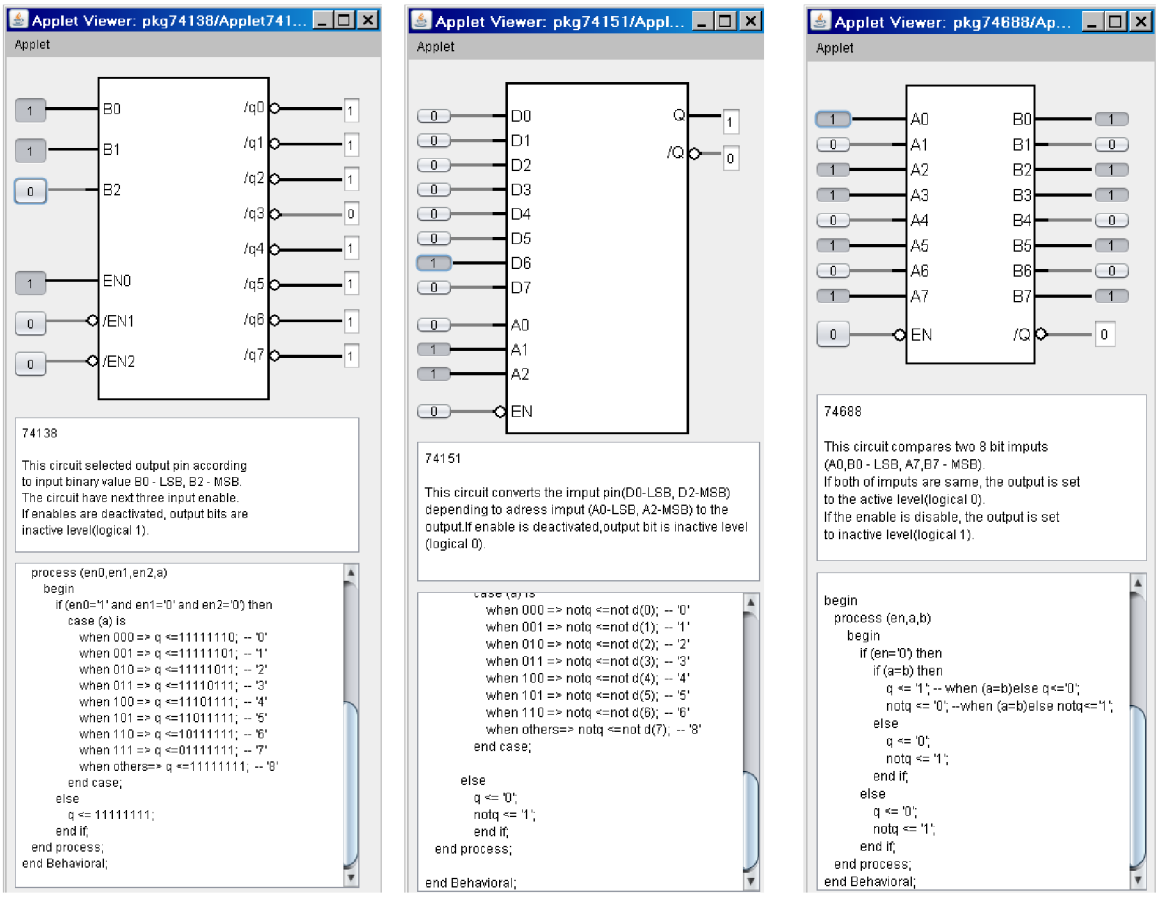
Z důvodu optimalizace appletů i pro starší verze Java Development Kitu, je nutno nastavit následující: v položce ***File*** otevřete ***Project Properties***. V levém sloupci vyberte ***Sources*** a poté nastavte ***Source/Binary Format: JDK 6***. Dále je potřeba implementovat knihovnu. V levém sloupci vyberte ***Libraries*** v záložce ***Compile*** klikněte na ***Add Library*** a vyberte knihovnu ***Absolute Layout***.

Dále je nutno dané applety podepsat a povolit jim Web Start, aby bylo například možno kopírovat do clipboardu [7]. Opět otevřete ***Project Properties***. V levém sloupci vyberte ***Web Start*** a zaškrtněte kolonku ***Enable Web Start***, dále zaškrtněte ***Applet descriptor.*** Pro podepsání daného appletu zmáčkněte tlačítko ***Customize…*** a zde vyberte možnost Self-sing by generated key. Poté stiskněte tlačítko ***OK***. Dále klikněte pravým tlačítkem na daný projekt a vyberte možnost ***Clean and Build.***

5.1 JAVA Applet - Digitální obvody

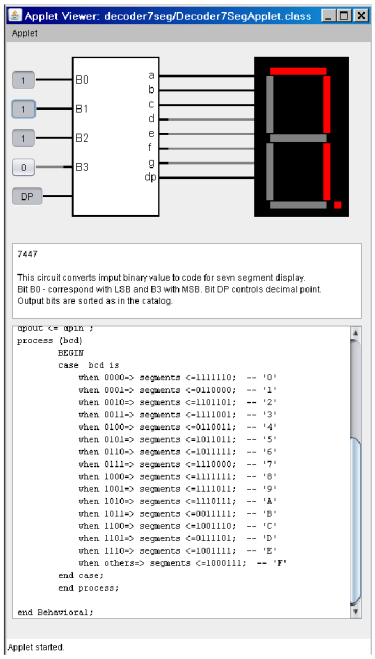
Java applety digitálních obvodů simulují funkce integrovaných obvodů a obvodu synchronního čítače zapojeného pomocí JK klopných obvodů. Každý applet dále obsahuje dvě textové pole. V jednom poli je jednoduše vysvětlen princip každého obvodu. Ve druhém textovém poli je funkce daného logického obvodu naprogramovaná v jazyku HDL, pro prostředí Xilinx ISE Design Suite. Mezi integrované simulované obvody patří:

- 74138 – podle binární hodnoty na adresových vstupech se na jednom z osmi výstupů objeví aktivní logická úroveň obvykle log. 0. Obr. 5.1a.
- 74151 – osmi-vstupový digitální multiplexor – selektor dat. Podle binární hodnoty na adresových vstupech se na výstupu objeví logická hodnota, která je přivedena na daný datový vstup na který odkazuje adresa. Obr. 5.1b.
- 74688 – porovnává dvě osmibitová čísla přivedená na datové vstupy. Jsou-li obě čísla shodná, na výstupu se objeví aktivní logická úroveň obvykle log. 0. Tento obvod v podstatě vykonává logickou funkci XOR. Obr. 5.1c.
- 7447 – obvod, který převádí binární kód přivedený na vstup obvodu na kód vhodný pro interpretaci dekadických čísel na sedmi-segmentovém displeji
- JK Synchronní čítač – ukázka vnitřní zapojení ho zapojení synchronního čítače Obr. 4.1.

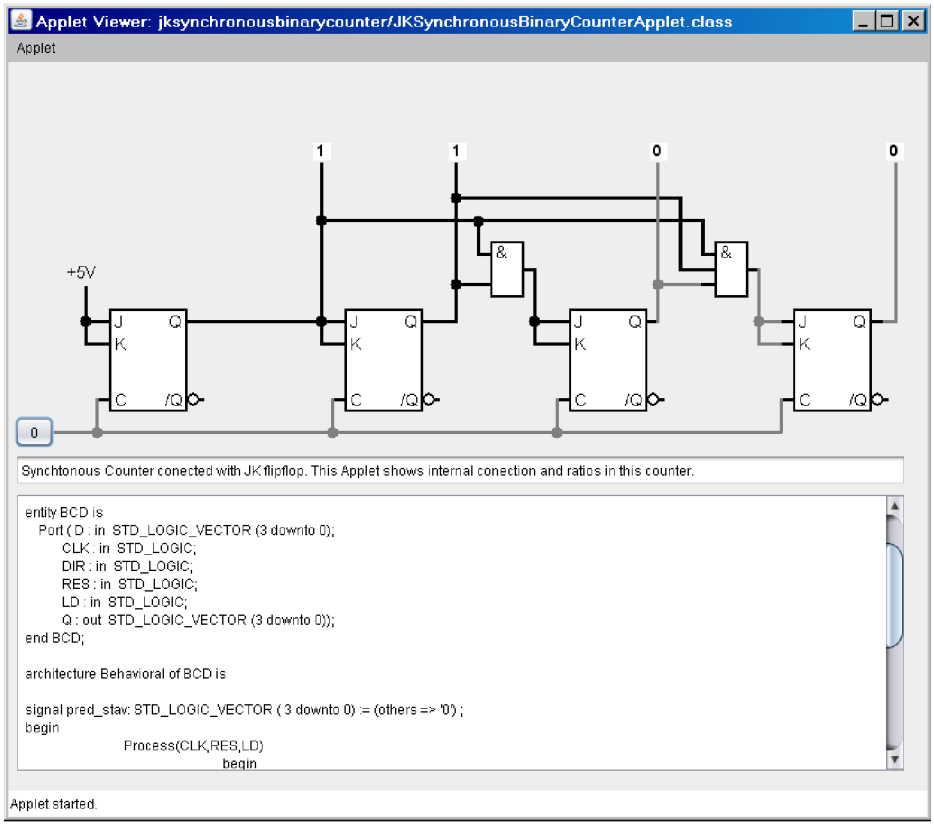


a) b) c)

Obr. 5.1: a) Applet obvodu 74138, b) Applet obvodu 74151, c) Applet obvodu 74688



Obr. 5.2: Applet obvodu dekodéru na sedmi-segmentový displej



Obr. 5.3: Applet obvodu vnitřního zapojení synchronního čítače vpřed na nástupnou hranu z klopných obvodů JK

5.2 JAVA Applet – Karnaughovy mapy

Applet Karnaughovy mapy je vytvořen pro minimalizování logických funkcí zadaných prostřednictvím vizualizace Karnaughovy mapy. Na výběr jsou tři možnosti velikosti map, mapa pro dvě, tři a čtyři vstupní proměnné. Vykreslování grafického prostředí pro dané velikosti Karnaughových map tzn.: počty tlačítek a vykreslení příslušného popisku má na starosti hlavní smyčka: `public void paint(Graphics g)`

Dále si uživatel může navolit tvar výstupní minimalizované funkce a to ve tvaru SOP – sum of products nebo POS – ve tvaru product of sums. Stisk každého tlačítka pak generuje hodnoty dané matice mpos nebo msop – funkce `private void termXActionPerformed(java.awt.event.ActionEvent evt)`.

Výsledek je vepsán do dvou řádků ve spodní části appletu. První řádek presentuje minimalizovanou logickou funkci v „textovém tvaru“, druhý řádek obsahuje stejnou minimalizovanou logickou funkci interpretovanou v jazyce HDL. Výpis výsledku v příslušném tvaru vrací funkce `public void Result(Vector <String> ResTerms)`.

Jako výpočetní jádro programu je použit Quine-McCluskeyho algoritmus s S. R. Petrickovou modifikací [6]. Minimalizace vstupní logické funkce proběhne po stisku GENERATE, kdy se vykoná funkce `private void generateActionPerformed` viz str.: 17. Tato funkce vezme příslušnou matici (msop nebo mpos) a převede její obsah – dekadická čísla na binární hodnoty. Tyto binární hodnoty doplní znakem 0 před binární číslo na délku N, kde N = počet vstupních proměnných. O výsledku rozhodne podmínka na konci tohoto algoritmu, která určí, zdali jsou v matici aktivní nebo neaktivní všechny hodnoty (tlačítka), pak se vypíše příslušný výsledek F = 1 nebo F = 0. Je-li jinak, volají se funkce algoritmu `firstImplicants()` a `minResult();`. Tyto funkce, za využití podfunkcí (viz níže), vrátí minimalizovaný tvar vstupní funkce. Některé z podfunkcí jsou převzaty a vhodně modifikovány [9]. Některé podfunkce dále volají vlastní funkce, které nejsou při popisu algoritmu uvedeny, protože jejich funkce nemá pro pochopení algoritmu zásadní vliv. Tyto funkce zpravidla porovnávají například znaky nebo vracejí počet jedniček v každém termu.

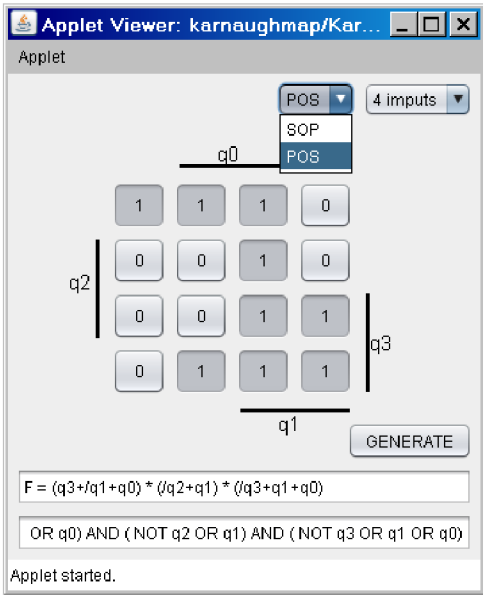
Pozn.: Jednotlivé funkce a podfunkce jsou součástí zdrojového kódu java appletu Karnaugh Map - příloha B.5:

*\KarnaughMap\src\ karnaughmap\KarnaughMapApplet.java

```
private   void   generateActionPerformed(java.awt.event.ActionEvent
evt) {
        minResult.setText("");
        takeTerm= new Vector <String>();
        implTab=new ArrayList[5][5];
        resTermVect= new Vector <String>();
        ResTerm = "";
        ResTermHDL = "";
        String reg="";

            for(int i=0;i<row;i++){
                for(int j=0;j<column;j++){
                    switch(functionShape.getSelectedIndex()){
                    case 0://for sop
                        num=numsop;
                        if(msop[i][j]!=20){
                            reg=Integer.toBinaryString(msop[i][j]);
                            if(reg.length()!=impVar){
  //while lenght of reg==num of input variables adds zeros
                                while(reg.length()!=impVar){
                                reg="0"+reg;
                                }
                            }
                            takeTerm.add(reg);
                        }break;
                    case 1://for pos
                        num=numpos;
                        if(mpos[i][j]!=20){
                            reg=Integer.toBinaryString(mpos[i][j]);
                            if(reg.length()!=impVar){
                                while(reg.length()!=impVar){
                                reg="0"+reg;
                                }
                            }
                            takeTerm.add(reg);

                        }break;
                    }
                }
            }
    if(takeTerm.size()==column*row){//term number == nuber of terms
                ResTerm="1";
                ResTermHDL="'1'";
                minResult.setText("F = "+ResTerm);
                minResultHDL.setText("F = "+ResTermHDL);
    }else if(takeTerm.size()==0){
                ResTerm="0";
                ResTermHDL="'0'";
                minResult.setText("F = "+ResTerm);
                minResultHDL.setText("F = "+ResTermHDL);
    }else{
                firstImplicants();
                minResult();
                minResult.setText("F = "+ResTerm);
                minResultHDL.setText("F = "+ResTermHDL);
            }
}
```



Obr. 5.4: Applet Karnaughovy mapy

5.2.1 Quine-McCluskeyho algoritmus

Metoda byla navržena filozofem W. O. Quinem a profesorem elektrotechniky a informatiky E. J. McCluskeym. Metoda využívá tří Booleovských zákonů. Zákonu distributivního, o opakování a o vyloučení třetího, z tohoto důvodu je nutno, aby daná funkce byla ve tvaru ÚDFN – úplném součtovém tvaru. Tato metoda je jednou z prvních metod vhodné k implementaci na počítači. Pro pochopení tohoto algoritmu si uvedeme jednoduchý příklad, ve kterém jsou objasněny jednotlivé funkce algoritmu.

Příklad 5.1.:

Minimalizujte funkci:

$$f = \overline{q_3}\overline{q_2}\overline{q_1}q_0 + \overline{q_3}\overline{q_2}q_1\overline{q_0} + \overline{q_3}q_2\overline{q_1}\overline{q_0} + q_3\overline{q_2}\overline{q_1}q_0 + q_3\overline{q_2}q_1\overline{q_0} + q_3\overline{q_2}q_1q_0 + q_3q_2\overline{q_1}q_0 + q_3q_2\overline{q_1}q_0 + q_3q_2q_1\overline{q_0} + q_3q_2q_1q_0$$

První krok: Převedeme výrazy do binární posloupnosti, kde $\overline{q_x} = 0$ a $q_x = 1$.

$$f = 0001 + 0010 + 0100 + 1001 + 1010 + 1011 + 1100 + 1101 + 1110 + 1111$$

Druhý krok: Rozdělení termů do skupin podle počtu logických jedniček (nul).

0001; 0010; 0100

1001; 1010; 1100

1011; 1101; 1110

1111

podfunkce:

```
public  ArrayList<Term>[][] termTable(Vector <String> tempRes)
```

Třetí krok: Srovnání dvojic ze sousedních skupin, které se liší maximálně v jednom bitu. Výsledné výrazy zapíšeme v nové iteraci, kde je opět rozdělíme do skupin podle počtu logických jedniček (nul). Rozdílný bit označíme „_“. Srovnání pro 1. a 2. skupinu.

_001; _010; _100

1001; 1010; 1100

1011; 1101; 1110

1111

podfunkce:

```
public  void makepair()
```

Čtvrtý krok: provedeme stejný postup jako ve třetím kroku, ale pro 2. a 3. skupinu.

_001; _010; _100

10_1; 1_01; 101_; 1_10; 110_; 11_0

1011; 1101; 1110

1111

podfunkce:

```
public  void makepair()
```

Pátý krok: provedeme stejný postup jako ve třetím kroku, ale pro 3. a 4. skupinu.

_001; _010; _100

10_1; 1_01; 101_; 1_10; 110_; 11_0

1_11; 11_1; 111_

podfunkce:

```
public  void makepair()
```

Šestý krok: provedeme stejný postup jako ve třetím kroku, ale pro 2. a 3. skupinu

Výrazy (_001; _010; _100) nelze dále zjednodušovat, nazýváme je primární implikanty.

_001; _010; _100

11_; 1_1_; 1__1

podfunkce:

```
public  void makepair()
```

Protože není nadále co zjednodušovat, algoritmus končí a součet výrazů je: $f = \overline{q_2 q_1} q_0 + \overline{q_2} q_1 \overline{q_0} + q_2 \overline{q_1 q_0} + q_3 q_2 + q_3 q_1 + q_3 q_0$

Kdybychom tutéž funkci minimalizovali pomocí Karnaughovy mapy dostali bychom menší počet termů ve výsledku než za použití Quine-McCluskeyho algoritmu. Proto byla navržena modifikace tohoto algoritmu S. R. Petrickem, která odstraní redundanci ve výsledku.

Výrazy ve výsledné funkci, kterou jsme dostali při použití Quine-McCluskeyho algoritmu, se nazývají minimální implikanty (prime implicants). Výraz *g* je implikantou výrazu *f*, pokud je každá proměnná výrazu *g* proměnnou výrazu *f*, a jestliže funkce implikace $g \rightarrow f$ nabývá všude hodnoty 1. Implikace nabývá hodnoty 1 pouze tehdy, když $g \leq f$ (přitom musíme předpokládat, že logická 0 je menší než logická 1 podobně jak v číselné algebře). Z této definice plyne, že *g* je implikantou *f* jen tehdy, když platí $g \leq f$.

Minimální implikantou logické funkce jsou výrazy, které splňují podmínku, že pokud vypustíme kterýkoliv výskyt kterékoliv proměnné negované nebo bez negace, dostaneme součin, který není implikantou daného výrazu. Jinak řečeno, minimální implikanta se nedá již dále zjednodušovat. Funkce, kterou jsme tedy dostali, se skládá z minimálních implikant, ale nevíme, zda se jedná o minimální disjunktivní formu.

5.2.2 Petrickova modifikace Quine-McCluskeyho algoritmu

K maximálnímu minimalizování logické funkce nám slouží S. R. Petrickova metoda.

Sedmý krok: Z výsledné funkce, kterou jsme dostali užitím Quine-McCluskeyho metody, sestavíme tabulku (viz Tab. 5.1), která má počet řádků rovný počtu všech termů v ÚNDF. Počet sloupců je roven počtu termů, které obsahuje výsledná funkce, kterou jsme dostali po použití Quine-McCluskeyho algoritmu. Do průsečíků řádků se sloupci vložíme symbol 1, jestliže minimální implikanta je obsažená ve výrazu uvedeném v daném řádku.

Tab. 5.1: Tabulka minimálních implikant.

	q_3q_2	q_3q_1	q_3q_0	$q_2\overline{q_1q_0}$	$\overline{q_2}q_1\overline{q_0}$	$\overline{q_2q_1}q_0$
0001						1
0010					1	
0100				1		
1001			1			1
1010		1			1	
1011		1	1			
1100	1			1		
1101	1		1			
1110	1	1				
1111	1	1	1			

Tabulku minimálních implikant můžeme nadále zjednodušit, ale musí být splněno to, že 1 ve sloupcích musí být rozložena mezi všechny řádky. Pokud by totiž jeden řádek neobsahoval ani jednu 1, pak by součet uvažovaných implikant neodpovídal dané funkci. Pakliže se v daném řádku nachází pouze jedna 1, implikantu v příslušném sloupci musíme zapsat do výsledné funkce. Implikanty, které nemůžeme vynechat, označujeme jako essential prime implicants.

podfunkce:

```
public  int essencImplicant(int[][] minTab)
```

Osmý krok - označíme všechny sloupce, které jsou (jakožto mintermy) pokryty označenými esenciálními primárními implikanty, tj. mají 1 v některém označeném řádku všechny označené řádky a sloupce z tabulky odstraníme. Dostaneme tak redukovanou tabulku minimálních implikant(viz Tab. 5.2).

Pozn.: platí-li pro nějaké dva různé sloupce c a d vztah c ≤ d: (zde porovnáváme vektory po složkách) odstraníme sloupec d.

Tab. 5.2: Redukovaná tabulka minimálních implikant.

	q_3q_2	q_3q_1	q_3q_0
1011		1	1
1101	1		1
1110	1	1	

Z tabulky 5.2 je zřejmé, že jakékoliv 2 implikanty pokrývají f. Výsledná funkce bude mít tedy tvar:

$$f = \underline{\underline{\overline{q_2q_1}q_0 + \overline{q_2}q_1\overline{q_0} + q_2\overline{q_1q_0} + q_3q_2 + q_3q_1}}$$

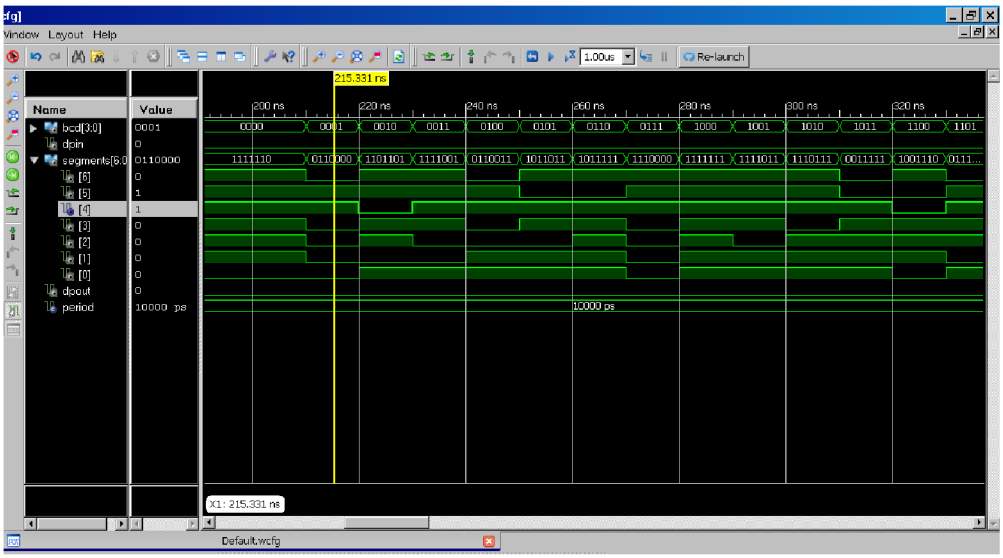
podfunkce:

```
public  void minResult()
```

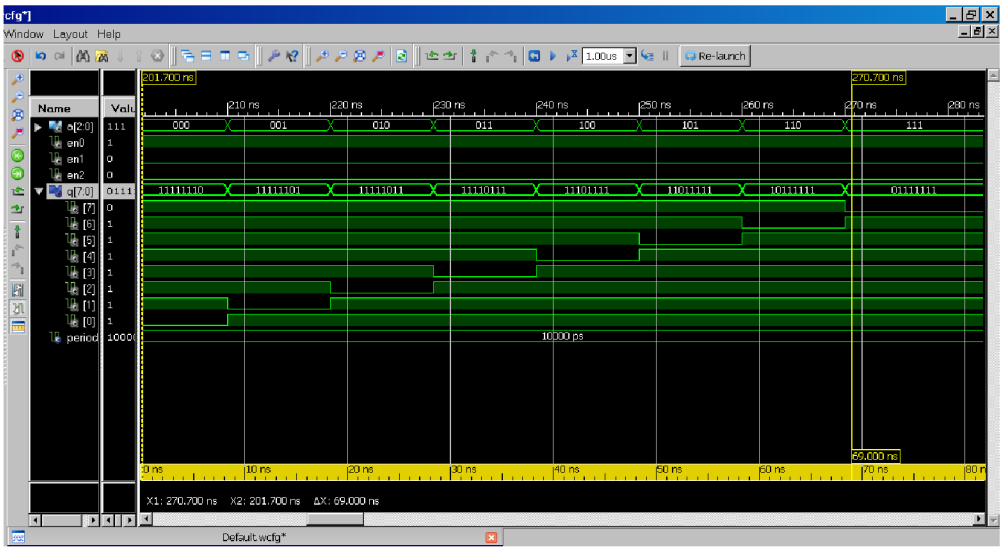
6 SIMULACE DIGITÁLNÍCH OBVODU V ISE XLINX DESIGN SUITE

Součástí naprogramovaných java appletů je textové pole s kódem v jazyce HDL, který popisuje funkci daného obvodu. Po importu tohoto kódu do prostředí ISE Xilinx Design Suite [10] a nastavení příslušných testovacích profilů (viz příloha C - *\"název obvodu"\stimulus.vhd), je možno ověřit funkce jednotlivých obvodů prostřednictvím simulačního programu v počítači.

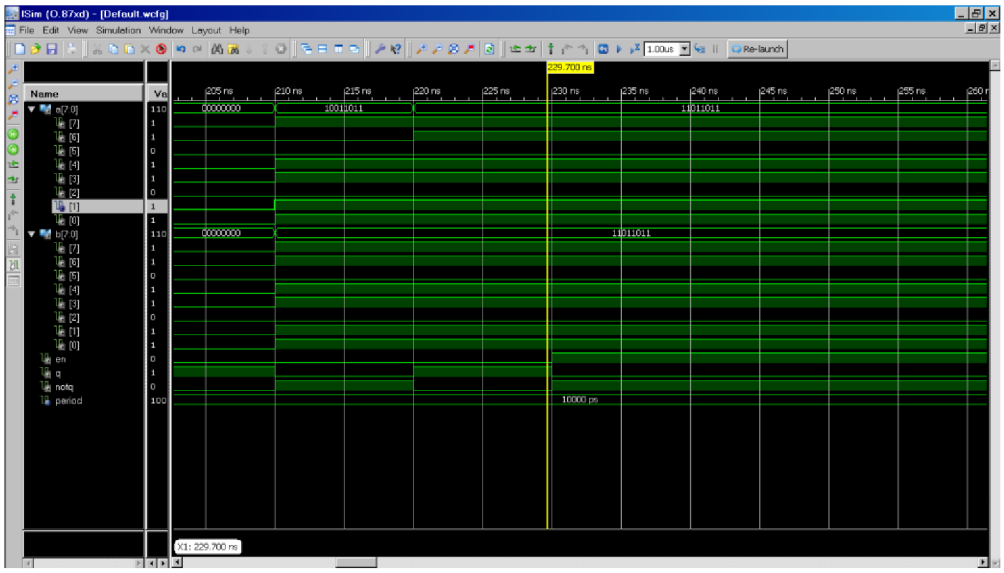
Výsledky simulací jsou uvedeny v následujících obrázcích (viz Obr 6.1, 6.2, 6.3, 6,4, 6.5).



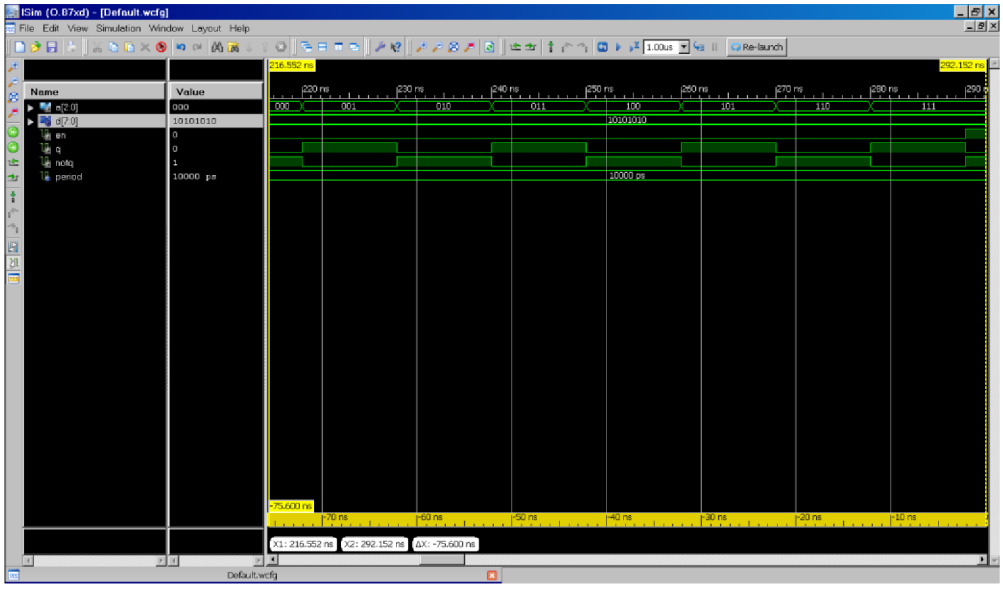
Obr. 6.1: Simulace obvodu 7447, dekodéru pro sedmi-segmentový displej.



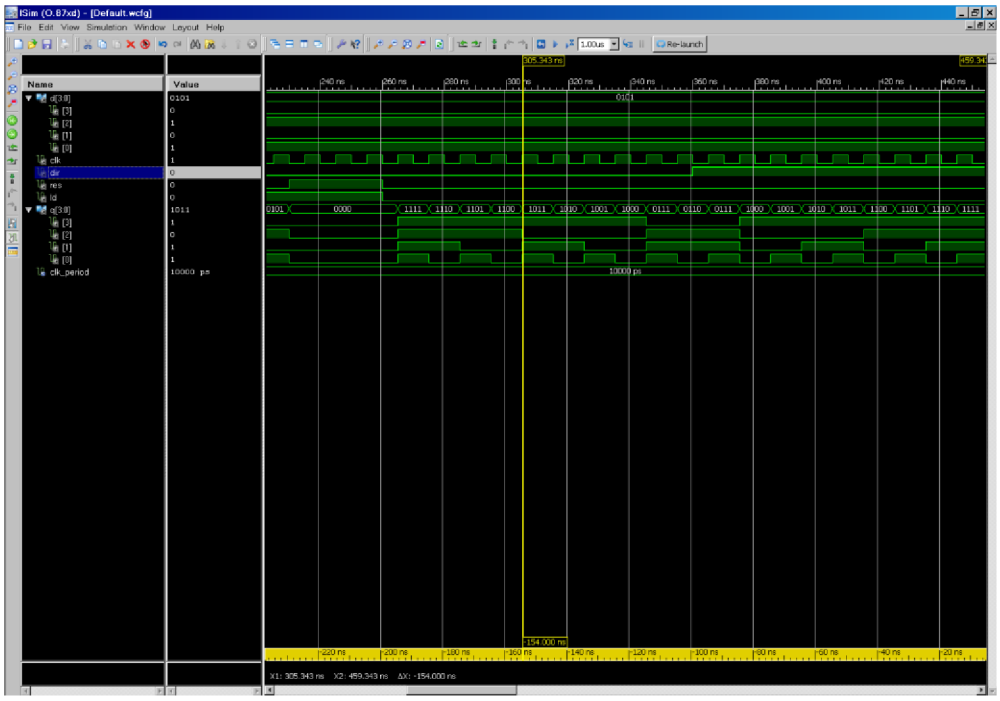
Obr. 6.2: Simulace obvodu 74138.



Obr. 6.3: Simulace obvodu 74688, digitálního komparátoru.



Obr. 6.4: Simulace obvodu 74151, multiplexoru.



Obr. 6.5: Simulace obvodu synchronního čítače s nastavitelnou počáteční hodnotou a směrem čítání.

7 ZÁVĚR

V této bakalářské práci se mi podařilo vytvořit zadání laboratorních úloh zaměřené na syntézu a číslicových systémů a jejich vlastností. Hlavní předností laboratorních úloh je využití moderního výukového systému RC 2000, který vyniká svojí názorností při realizaci návrhu kombinačně logických funkcí.

V úloze zaměřené na syntézu a realizaci synchronních a asynchronních čítačů se studenti seznámí s realizací návrhu těchto obvodu pomocí klopných obvodů i integrovaných čítačů. Správnost jejich zapojení si poté mohou ověřit na sedmi-segmentovém displeji.

V úloze zaměřené na detekci hazardů u logických obvodů se studenti seznámí s detekcí hazardů s použitím osciloskopu.

V úloze zaměřené na realizaci logických funkcí pomocí speciálních kombinačních obvodů se studenti seznámí s analýzou těchto obvodů pomocí logického analyzátoru, jež je součástí systému RC 2000. Při realizaci laboratorních úloh jsem nezaznamenal žádný problém, v úloze zaměřené na analýzu hazardů jsem pouze musel využít osciloskopu k detekci těchto stavů z důvodu nedostatečné vzorkovací frekvence univerzální vstupně/výstupní jednotky.

V této bakalářské práci se mi dále podařilo naprogramovat java applety, které simulují funkce některých digitálních obvodů běžně používaných v číslicové technice. Tyto applety jsou umístěny na internetu, takže si každý student může jednotlivé funkce daných obvodů vyzkoušet. Dále si studenti mohou nasimulovat tyto obvody v prostředí ISE Xilinx Design Suite.

Další naprogramovaný applet simuluje minimalizace logické funkce pomocí Karnaughových map. Pro minimalizování funkce je použit Quine-McCluskeyho algoritmus s Petrickovou modifikací.

Při programování appletů jsem se setkal s problémem při kopírování textu z textového pole do clipboardu. U starších verzí balíku JDK s tímto nebyl problém, ale u novějších balíků je toto z důvodů bezpečnosti zakázáno. Řešení je uvedeno výše.

LITERATURA

[1] KOLOUCH, J., BIOLKOVÁ, V. *Impulzová a číslicová technika.* Elektronické skriptum. Brno: FEKT VUT v Brně, 2009.

[2] VRBA, R., LEGÁT, P., KUCHTA, R., MIKEL, B. *Digitální obvody a mikroprocesory.* Elektronické skriptum. Brno: FEKT VUT v Brně, 2006.

[3] FRÝZA, T. Stánky předmětu Impulzová a číslicová technika [online]. 2012 – [cit. 5. prosince 2012]. Dostupné na WWW: http://www.urel.feec.vutbr.cz/~fryza/.

[4] Stránky firmy RC DIDACTIC [online]. 2012 – [cit. 5. prosince 2012]. Dostupné na WWW: http://www.rcdidactic.cz/cz/.

[5] Stránky Indiana Univerzity [online]. 2006 – [cit. 5. prosince 2012]. Dostupné na WWW: http:// http://www.cs.indiana.edu/classes/b441-sjoh/.

[6] HORKÝ, M. *Minimalizace logických funkcí.* Bakalářská práce. Brno: FEKT VUT v Brně, 2009.

[7] Stránky firmy Oracle [online]. 2011 – [cit. 20. května 2013]. Dostupné na WWW: https://blogs.oracle.com/kyle/entry/copy_and_paste_in_java

[8] Stránky firmy NetBeans [online]. 2011 – [cit. 22. května 2013]. Dostupné na WWW: https://netbeans.org/

[9] Stránky Java forum Stackoverflow [online]. 2012 – [cit. 22. května 2013]. Dostupné na WWW: http://stackoverflow.com

[10] Stránky firmy XILINX [online]. 2011 – [cit. 22. května 2013]. Dostupné na WWW: http://www.xilinx.com/

SEZNAM PŘÍLOH

A LABORATORNÍ ÚLOHY

A.1 Hazardy

HAZARDY	BICT				Datum měření:							2013		Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

XX HAZARDY

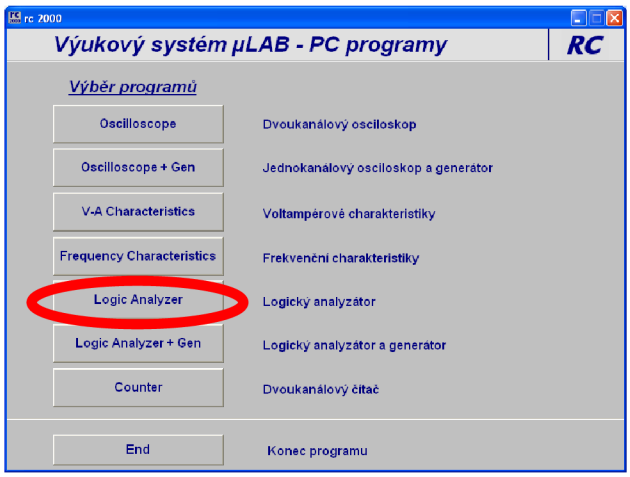
XX.1 Zadání

1. Zapojte elementární struktury pro demonstraci statického hazardu s logickými členy NAND. Zobrazte průběhy napětí na jejich výstupu a na osciloskopu detekujte přítomnost parazitního impulzu - glitche.
2. Demonstrujte dynamický hazard. Využijte k tomuto logického členu NAND a obou zapojení z předchozí úlohy. Zobrazte na osciloskopu.
3. Navrhněte obvod pro demonstraci dynamického hazardu. Využijte k tomuto logické členy. Vycházejte z následujícího obrázku. Kde $\tau 1 < \tau 2 < \tau 3$ a navrhněte klopný obvod KLO3 znáte-li následující průběhy f1(a), f2(a), f3(a) a f(a).
4. Z tabulky hodnot navrhněte Karnaughovu mapu. Vyznačte na ní místo hazardu (vycházejte z nejjednoduššího možného tvaru výsledné funkce) a odvoďte výstupní funkci v minimálním součtovém tvaru. Výslednou funkci zapojte pomocí obvodu NAND a proveďte analýzu hazardu na osciloskopu, poté hazard odstraňte a opět ověřte na osciloskopu.

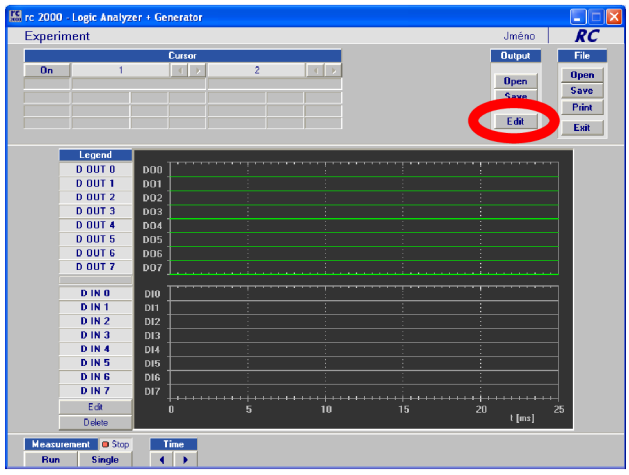
XX.2 Pokyny k zadání

1. Pomocí výukového systému RC Didactic zapojte návrh, elementární struktury demonstraci statického hazardu s logickými členy NAND. Z důvodu zřetelnějšího projevu glitche při větším zpoždění, využijte v logickém obvodu pro spodní větev obvodu všechny tři hradla zapojená do série. Měření proveďte na obvodu 7400 jak pro novou verzi RC 2000 (magnetické kartičky), tak i pro starší verzi s integrovaným obvodem 7400. Jednotlivé zapojení NEROZPOJUJTE! Připojte Analog and data unit k PC a spusťte program RC 2000, z nabídky vyberte Logic analyzer + Gen. V obslužném programu v bloku OUTPUT stiskněte EDIT a poté v bloku MODE stiskněte 1 BIT. Myší nakreslete průběh jedné nástupné hrany. Časovou osu nastavte na nejkratší interval a stisknutím LEAVE EDITOR se vraťte zpět. Propojte výstup DIGITAL OUTPUT se vstupem měřeného obvodu a na vstup A&DDU připojte výstup měřeného obvodu. Stiskněte tlačítko RUN. Z důvodu rychlé změny použijte k detekci glitche osciloskop. Vstup i výstup měřeného obvodu zároveň připojte k digitálnímu osciloskopu. Vstup na kanál 1 a výstup na kanál 2. Na osciloskopu nastavte stiskem žlutého tlačítka ch1 menu a nastavte PROBE na 1x VOLTAGE a zkontrolujte, zda je toto nastaveno i na přepínači na

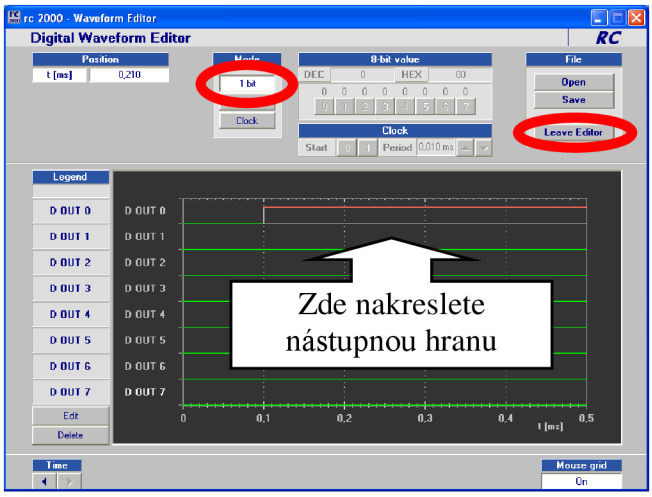
sondě osciloskopu. Pokud ne, učiňte tak, jinak nebudou odpovídat měřítka osy Y. Toto nastavení proveďte i pro kanál 2. Ve skupině tlačítek TRIGGER stiskněte SET TO 50%. Stiskněte tlačítko TRIGGER MENU a nastavte: TYPE-EDGE, SOURCE CHANNEL - CH1, SLOPE - RISSING, COUMPLING – DC.



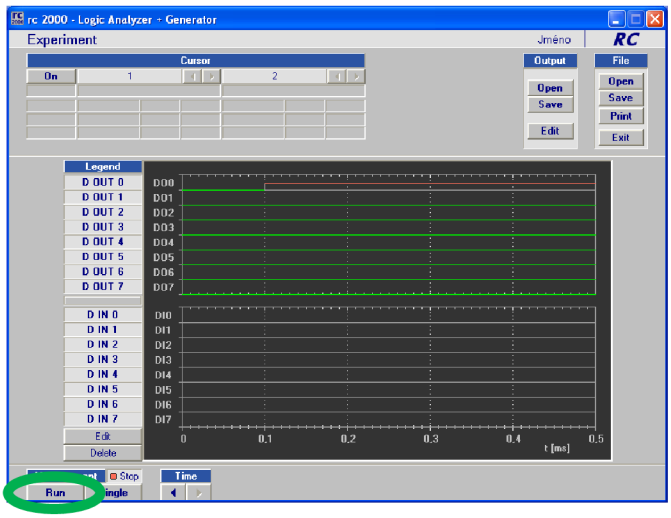
Obrázek 1 - výběr funkce logické jednotky



Obrázek 2 - výběr nastavení editoru výstupních signálů



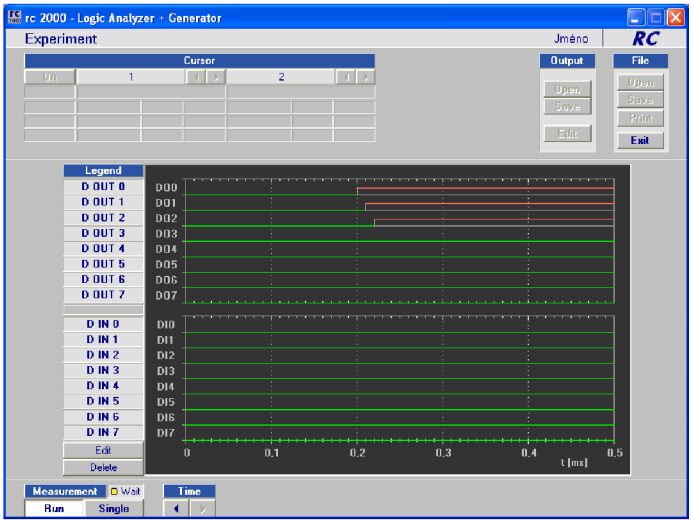
Obrázek 3 - nastavení náběžné hrany



Obrázek 4 - spuštění generátoru logických hodnot

2. Demonstrujte hazard. Využijte k tomuto logického členu NAND a obou zapojení z předchozí úlohy s obvody 7400. Na první vstup připojte výstup prvního obvodu 7400, na druhý vstup připojte výstup druhého obvodu 7400. Na osciloskopu zobrazte vzniklý hazard. Postup je stejný jako v předchozí úloze.

3. Zapojte obvod podle domácí přípravy. Pro ověření správné funkce obvodu přiveďte na jeho vstupy tři od sebe zpožděné nástupné hrany a ověřte, že na výstupu se objeví signál shodný se signálem f(a) ze zadání. K simulaci opět využijeme program RC 2000. Z nabídky funkcí A&DDU, vyberte Logic analyzer + Gen. V obslužném programu v bloku OUTPUT stiskněte EDIT a poté v bloku MODE stiskněte 1 BIT. Časovou osu nastavte na nejmenší hodnotu. Do prostoru pro časové průběhy D OUT 0 – D OUT 2 nakreslete myší jednotlivé průběhy, stejně jako v zadání. Vstupy měřeného obvodu připojte k jednotlivým výstupům D OUT 0 – D OUT 2. Opusťte editor časových průběhů stiskem tlačítka LEAVE EDITOR. Postup je stejný jako v bodě 1. Na osciloskopu zobrazte výsledný časový průběh napětí z výstupu měřeného obvodu. Poté místo výstupu D OUT 1 připojte reálně zpožděný signál po průchodu jedním invertorem a místo DOUT 2 připojte reálně zpožděný signál po průchodu pěti invertory. Dále postupujte stejným způsobem. Výsledný glitch zobrazte na osciloskopu a změřte dobu jeho trvání.



Obrázek 5 - ověření funkce obvodu pomocí uměle vygenerovaného zpoždění

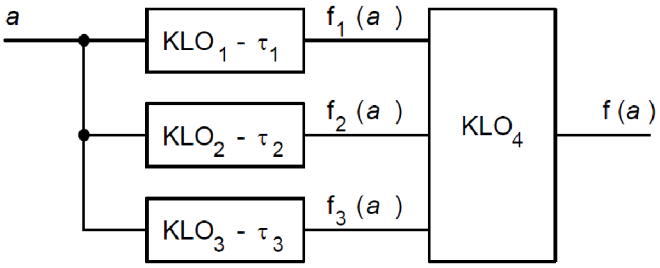
4. Funkci odvozenou v domácí přípravě zapojte pomocí obvodu NAND a proveďte analýzu hazardu na osciloskopu. Pro generování časových průběhů využijte A&DDU mód Logic analyzer + Gen. Z pravdivostní tabulky určete, při které změně vstupní hodnoty se projevuje hazard a příslušný průběh nastavte na generátoru stejně jako v předchozích úlohách.

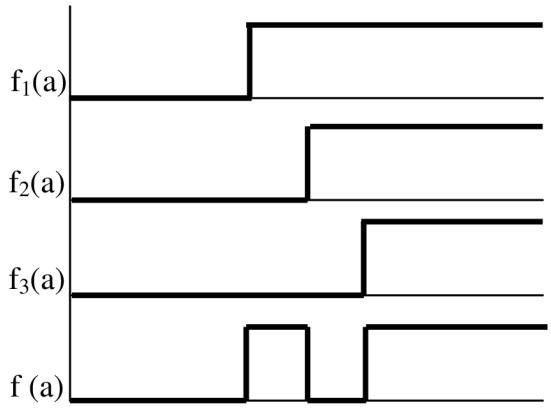
Obrázek 6 - nastavení časového průběhu při změně určité vstupní hodnoty

HAZARDY	BICT				Datum měření:						2013		Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

!!! Domácí příprava!!!

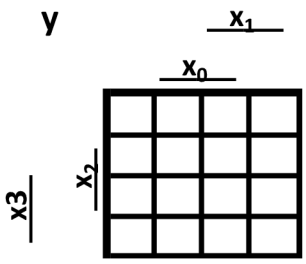
a) Navrhněte klopný obvod KLO_4 tak, aby při vstupních proměnných $f_1(a)$-$f_3(a)$ byl na výstupu průběh napětí f(a).





b) Z tabulky hodnot navrhněte Karnaughovu mapu. Vyznačte na ní místo hazardu (vycházejte z nejjednoduššího možného tvaru výsledné funkce) a odvoďte výstupní funkci v minimálním součtovém tvaru.

x3	x2	x1	x0	y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



XX.3 Teoretické poznatky

HAZARDY

Hazardy jsou stavy v kombinačně logických obvodech, při kterých v důsledku zpoždění jednotlivých obvodů může (nemusí) dojít ke vzniku parazitních impulzů na výstupu tohoto obvodu. Doba trvání těchto impulzů je především závislá na teplotě, velikosti napájecího napětí a na podobných vlivech, proto přesné zpoždění jednotlivých logických obvodů neznáme. V katalozích se pak udává jen hraniční hodnota. Parazitní impulzy, které se mohou objevit na výstupu kombinačně logických obvodů, označíme jako GLITCH. Hazardy můžeme rozdělit na dva základní typy a to hazardy dynamické a hazardy statické.

Statické hazardy se projevují u signálů, u nichž očekáváme stálou logickou úroveň, ale při změně sledované vstupní veličiny může na výstupu kombinačně logické funkce vzniknout parazitní impulz opačné polarity.

Dynamický hazard

Dynamické hazardy se projevují tehdy, očekáváme-li při změně vstupní veličiny změnu veličiny výstupní. Zpravidla očekáváme, že jedna změna vstupní veličiny způsobí jednu změnu výstupní veličiny. Je-li však v obvodu dynamický hazard, může se odezva výstupní veličiny skládat, to znamená, že při změně z jedné logické úrovně do druhé se mohou před ustálením logické hodnoty objevit parazitní impulzy, které mohou ovlivnit výslednou logickou funkci obvodu.

Statický hazard

Hazardy jsou nedílnou součástí návrhu kombinačních logických funkcí a pro jejich analýzu používáme dvou základních přístupů a to přístupu algebraického a přístupu za pomocí použití Karnaughových map. Jednotlivé přístupy k řešení hazardů a jejich vysvětlení naleznete ve skriptech[1].

Pro správnou funkci obvodu je nezbytné, aby tyto stavy byly co nejvíce potlačeny. K tomuto můžeme přistoupit několika způsob:

1. Zpožděním výstupního signálu a jeho použití, až po uplynutí určité doby, po které se výstup ustálí.
2. Použití synchronních obvodů, ve kterých se parazitní impulzy nevyskytují.
3. Je možno použít filtru RC s charakterem integračního článku, ale tento postup rozhodně nelze doporučit jako vhodný pro globální řešení hazardů v kombinačně logických obvodech. Tento postup se používá jen zřídka a v nouzi.

Při návrhu kombinačně logické funkce pomocí hradel lze hazard odstranit přidáním termu do výsledné logické funkce. Tohoto můžeme dosáhnout různými způsoby. Prvním způsobem je řešení hazardů pomocí Karnaughových map. Druhým způsobem řešení odstranění hazardů je algebraický přístup.

HAZARDY	BICT				Datum měření:					2013			Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

Úloha 1.

Pozn.: Do prázdných míst vlep obrázky vytištěné osciloskopem popr. dokresli a okótuj

Glitch u novější verze RC 2000

Glitch u novější verze RC 2000

	Novější verze RC 2000	Starší verze RC 2000
Doba trvání glitche [ns]		
Napěťová úroveň glitche [V]		

Úloha 2.

glitch vlivem různého zpoždění obvodů po průchodu členem NAND

Doba trvání glitche [ns]	
Napěťová úroveň glitche [V]	

Úloha 3.

glitch způsobený vygenerovaný zpožděním glitch způsobený reálným zpožděním

Glitch způsobený reálným zpožděním	Doba trvání glitche [ns]	Napěťová úroveň glitche [V]

Úloha 4.

hazard na sestupné hraně

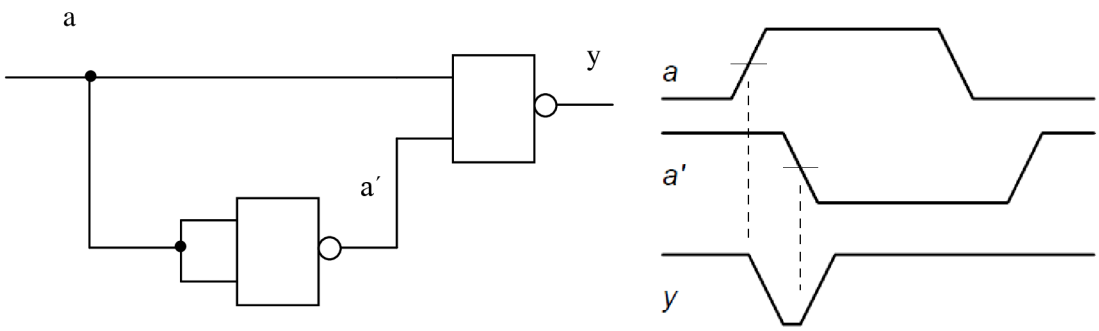
Doba trvání glitche [ns]	
Napěťová úroveň glitche [V]	

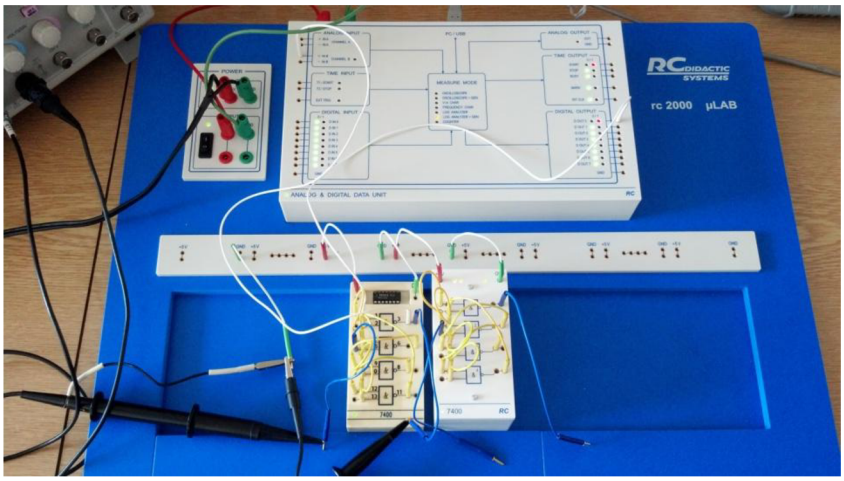
HAZARDY	BICT				Datum měření:						2013		Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

VYPRACOVÁNÍ – PRO UČITELE

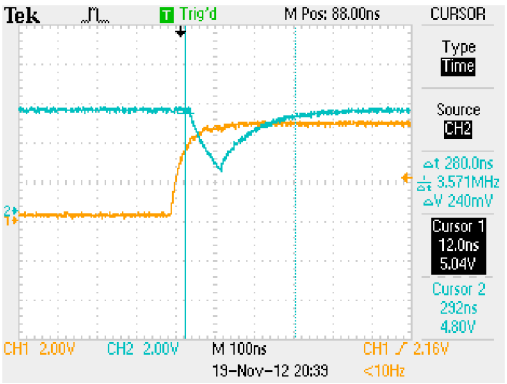
Hazardy:

1. Zapojte elementární struktury pro demonstraci statického hazardu s logickými členy NAND. Zobrazte průběhy napětí na jejich výstupu a na osciloskopu detekujte přítomnost parazitního impulzu - glitche.

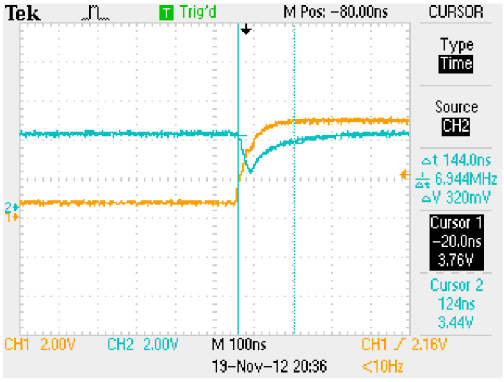




Obrázek 7 - zapojení úkolu 1

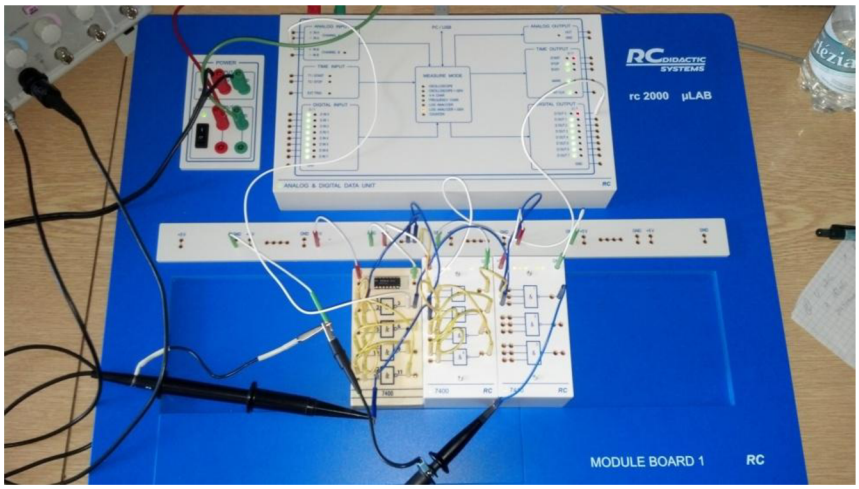


Obrázek 8 - glitch elementárního obvodu u novější verze RC 2000

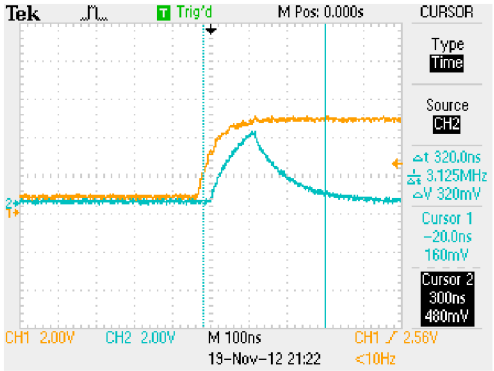


Obrázek 9 - glitch elementárního obvodu u starší verze RC 2000

2. Demonstrujte dynamický hazard. Využijte k tomuto logického členu NAND a obou zapojení z předchozí úlohy. Zobrazte na osciloskopu.

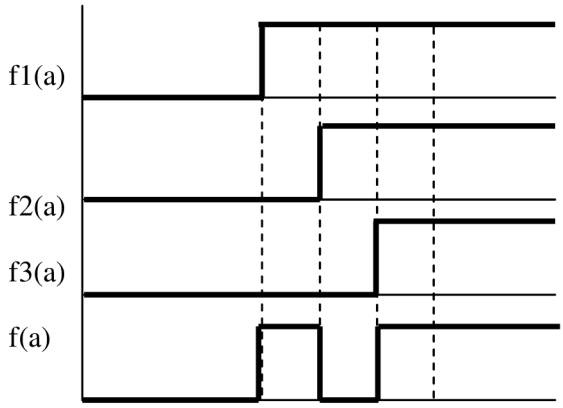


Obrázek 10 - zapojení úlohy 2



Obrázek 11 - glitch vlivem různého zpoždění obvodů po průchodu členem NAND

3. Navrhněte obvod pro demonstraci dynamického hazardu. Využijte k tomuto logické členy. Vycházejte z následujícího obrázku. Kde $\tau 1 < \tau 2 < \tau 3$ a navrhněte klopný obvod KLO3 znáte-li následující průběhy f1(a), f2(a), f3(a).

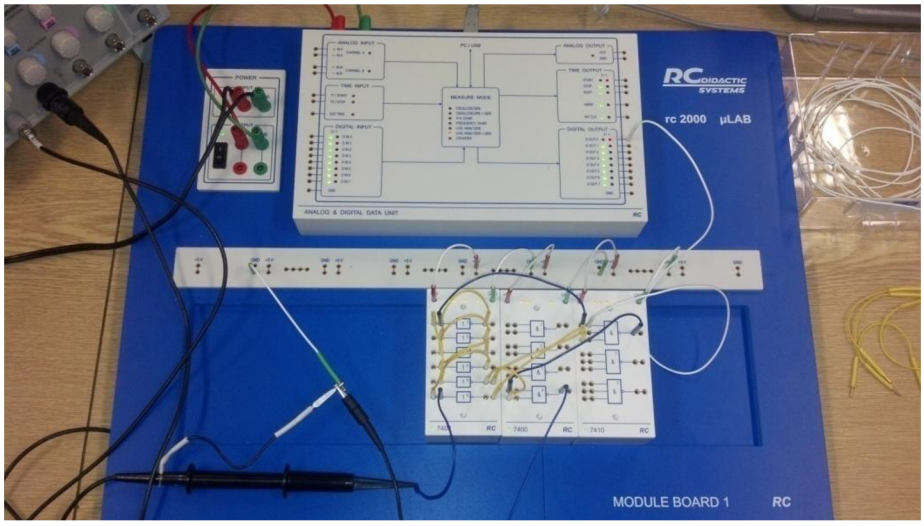


f3(a)	f2(a)	f1(a)	f(a)
0	0	0	0
0	0	1	1
0	1	0	x
0	1	1	0
1	0	0	x
1	0	1	x
1	1	0	x
1	1	1	1

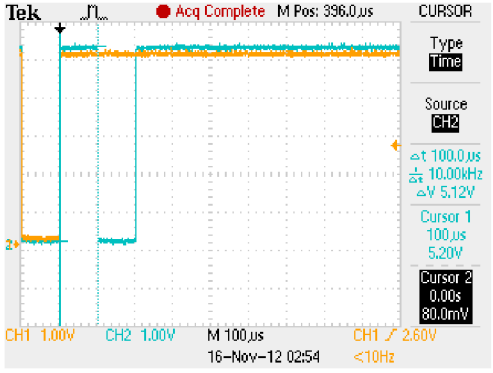
f(a)

f3(a) \ f2(a) f1(a)	0	1	0	x
x	x	x	1	x

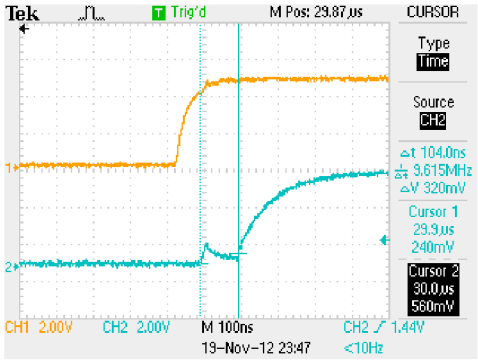
$$f(a)_{sop} = x_0\,\overline{x_1}\,\overline{x_2} + x_2 = \overline{\overline{x_0\,\overline{x_1}\,\overline{x_2}} \cdot \overline{x_2}}$$



Obrázek 12 - zapojení úlohy 2



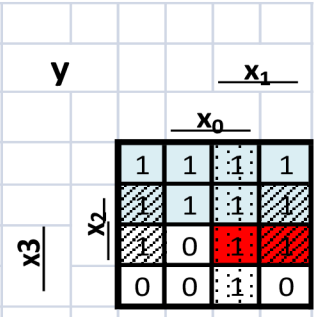
Obrázek 13 - glitch způsobený uměle vygenerovaný zpožděním pomocí RC 2000



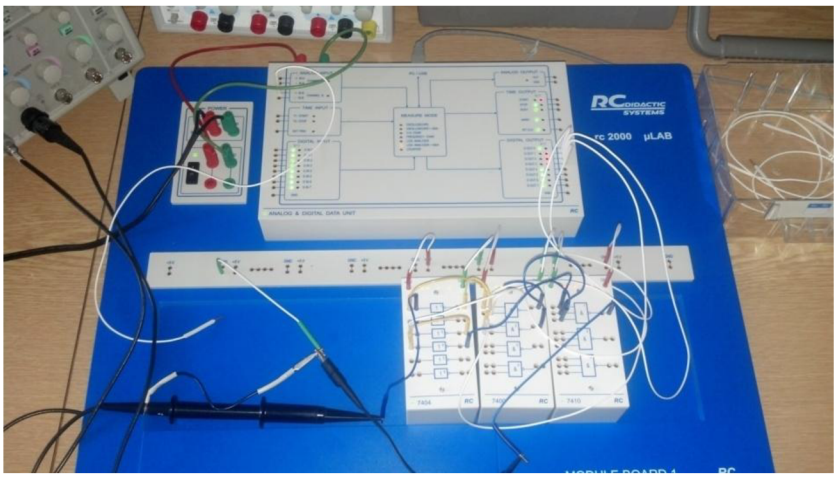
Obrázek 14 - glitch způsobený reálným zpožděním obvodu

4. Z tabulky hodnot navrhněte Karnaughovu mapu. Vyznačte na ní místo hazardu (vycházejte z nejjednoduššího možného tvaru výsledné funkce) a odvoďte výstupní funkci v minimálním součtovém tvaru. Výslednou funkci zapojte pomocí obvodu NAND a proveďte analýzu hazardu na osciloskopu, poté hazard odstraňte a opět ověřte na osciloskopu.

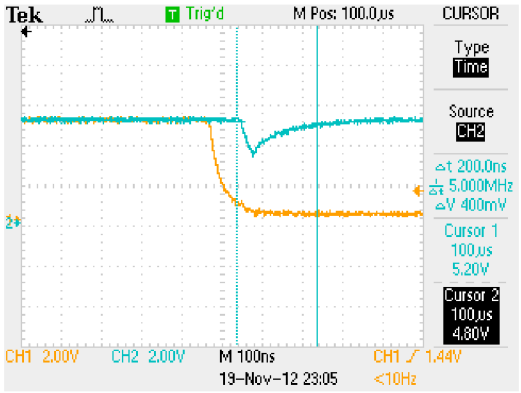
x3	x2	x1	x0	y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1



$$y = x_0x_1 + \overline{x_0}x_2 + \overline{x_3} = \overline{\overline{x_0x_1} \cdot \overline{\overline{x_0}x_2} \cdot \overline{\overline{x_3}}}$$



Obrázek 15 - zapojení úlohy 4



Obrázek 16 - hazard na sestupné hraně

A.2 Syntéza kombinačních logických funkcí pomocí speciálních kombinačních obvodů

SKLFPKO	BICT				*Datum měření:*							*2013*		*Příjmení a jméno:*
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

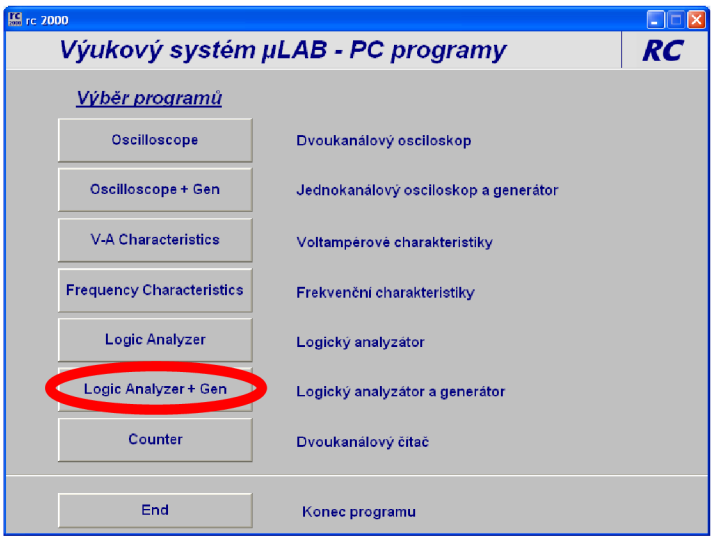
XX SYNTÉZA KOMBINAČNÍCH LOGICKÝCH FUNKCÍ POMOCÍ KOMBINAČNÍCH OBVODŮ

XX.1 Zadání

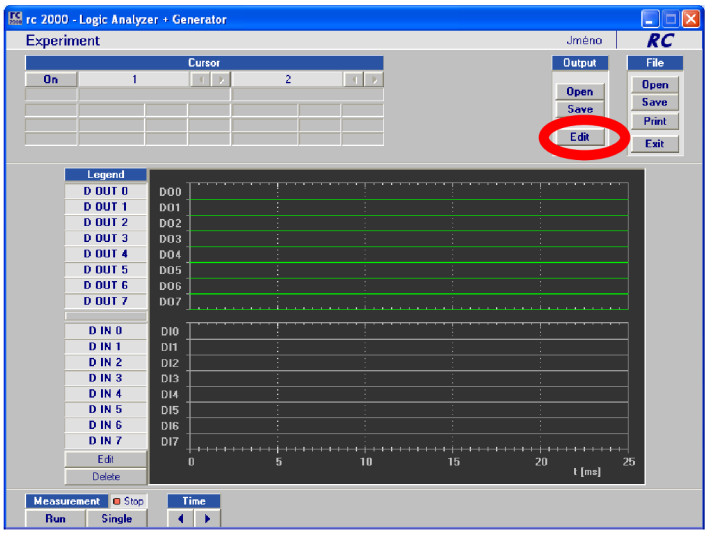
1. Navrhněte pravdivostní tabulku pro dvoubitovou sčítačku s výstupním bitem pro přenos do vyššího řádu. Pomocí Karnaughovy mapy minimalizujte výsledné kombinačně logické funkce a zapojte je pomocí obvodu NAND.
2. Pomocí dvojného čtyřbitového multiplexoru realizujte funkci nižšího bitu sčítačky (y_0). Z důvodu kontroly vyučujícím, proveďte eliminaci pro bity b_0, b_1, a_0. Zapojení navrhněte tak, aby multiplexor pracoval jako osmibitový.
3. Ověřte funkci dvoubitové sčítačky pomocí integrovaného zapojení 74283.

XX.2 Pokyny k zadání

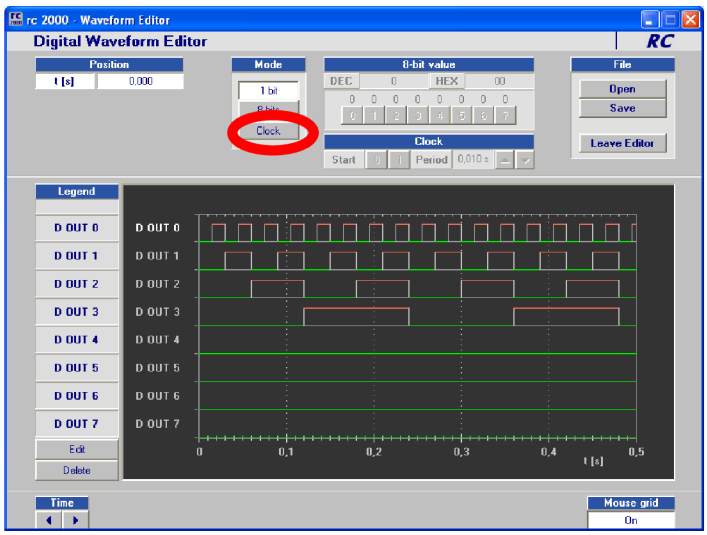
1. Pomocí výukového systému RC Didactic zapojte dvoubitovou sčítačku, jejíž výstupní logické funkce jste vypracovali v domácí přípravě. Z důvodu časově náročného celkového zapojení, zapojte pouze logickou funkci pro nejnižší bit b_0 a přenosový bit do vyššího řádu (carry bit) C. Připojte Analog and data unit k PC a spusťte program RC 2000, z nabídky vyberte Logic analyzer + Gen. V obslužném programu v bloku OUTPUT stiskněte EDIT a poté v bloku MODE stiskněte CLOCK. Periodu hodinového signálu pro D OUT 0 nastavte 30ms, pro D OUT 1 60ms, pro D OUT 2 120ms a pro D OUT 3 240ms. Časovou osu nastavte na 0,5s. Stisknutím LEAVE EDITOR se vraťte zpět. Propojte výstup DIGITAL OUTPUT se vstupem měřeného obvodu a na vstup A&DDU do bloku DIGITAL INTPUT připojte výstupy měřeného obvodu (bit y_0 a carry bit). Stiskněte tlačítko SINGLE. Výsledný průběh si uložte a ověřte jeho správnost pomocí tabulky z domácí přípravy. Výsledné zapojení NEROZPOJUJTE!



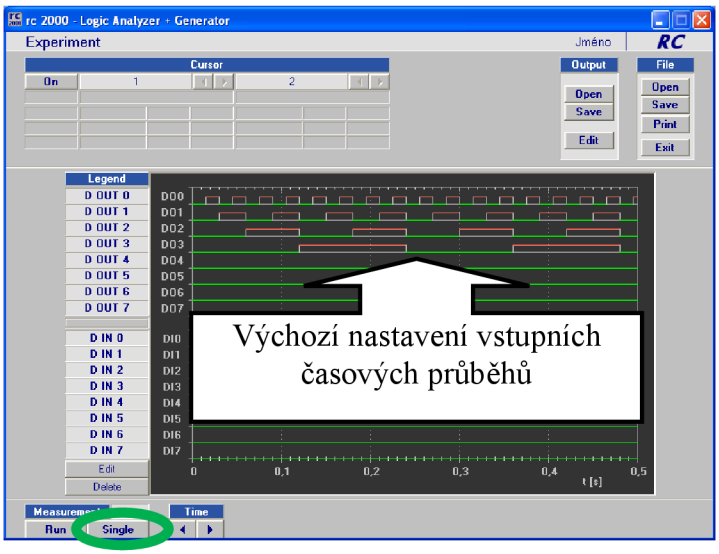
Obrázek 17 - výběr funkce logické jednotky



Obrázek 18 - výběr nastavení editoru výstupních signálů



Obrázek 19 - nastavení požadovaných hodinových průběhů



Obrázek 20 - spuštění generátoru a výsledné časové průběhy

2. Pomocí multiplexoru 74153 zapojte návrh funkce nižšího bitu sčítačky (y_0) u dvoubitové sčítačky, který jste vypracovali v domácí přípravě. Na příslušné vstupy obvodu opět přiveďte hodinový signál D OUT 0 - D OUT 3. Výstup obvodu připojte opět na digitální vstup jednotky A&DDU. Výsledný časový průběh porovnejte s časovým průběhem výstupní logické funkce pro bit C z předchozího bodu zadání. Obrázek výsledných časových průběhů si opět uložte pomocí funkce Print Screen.

3. Ověřte funkci dvoubitové sčítačky pomocí integrovaného zapojení 74258. Nastavení časových průběhů vstupních hodnot nechte stejné jako v předchozích úkolech, pouze na digitální vstupy logického analyzátoru připojte příslušné výstupy dvoubitové sčítačky. Nepoužité vstupní bity připojte na nízkou logickou úroveň. Spusťte simulaci pomocí tlačítka SINGLE.

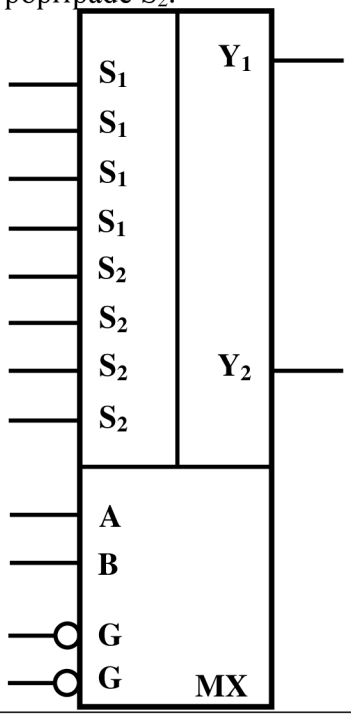
SKLFPKO	BICT				Datum měření:						2013		Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

!!! Domácí příprava!!!

c) Navrhněte pravdivostní tabulku pro dvoubitovou sčítačku s výstupním bitem pro přenos do vyššího řádu. Pomocí Karnaughovy mapy minimalizujte výsledné kombinačně logické funkce a upravte je do součinového tvaru pro zapojení z obvodů NAND.

č.	a1	a0	b1	b0	c	y1	y0
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							

d) Pomocí dvojného čtyřbitového multiplexoru realizujte funkci nižšího bitu sčítačky (y_0). Z důvodu kontroly vyučujícím, proveďte eliminaci pro bity b_0, b_1, a_0. Zapojení navrhněte tak, aby multiplexor pracoval jako osmibitový. Nápověda: Vstupy G_1 a G_2 aktivují čtyřbitový vstup S_1 popřípadě S_2.



XX.3 Teoretické poznatky

Realizací kombinačně logické funkce rozumíme sestavení schématu zapojení pomocí příslušných obvodů, které ze vstupních proměnných vytvoří výstupní proměnnou s požadovanou logickou funkcí. V praxi se pro realizaci kombinačně logické funkce často používá pouze jeden Integrovaný obvod. Tento se vyrábí sériově a najdeme ho v katalogu nebo se sériově nevyrábí a k realizaci funkce využijeme paměti (PROM) nebo obvody PLD. Přesto se v základu vždy vychází ze zápisu logické funkce pomocí tvaru SOP (Sum Of Products) nebo POS (Product Of Sums).

Nejčastější způsoby realizace kombinačně logických funkcí v číslicové technice jsou:

- pomoci integrovaných obvodů typu NAND, INVERT, NOR
- pomocí multiplexorů a demultiplexorů
- pomocí speciálních obvodů – převodníky kódu, násobičky, sčítačky
- pomocí pamětí PROM nebo programovatelných obvodů typu PLD

V této úloze využijeme k realizaci obvodů především obvodů typu NAND (součinový tvar funkce) a multiplexorů. Výhodou zapojení pomocí obvodů typu NAND je, že pokud nevyužijeme všechny členy NAND v jednom pouzdře při zapojení první logické funkce, můžeme tyto obvody využít i při zapojení následující logické funkce

Realizace pomocí obvodů typu NAND

Pro realizaci obvodů pomocí této metody vycházíme z Karnaughovy mapy do které zapíšeme hodnoty výstupní funkce. Poté tuto funkci zapíšeme v minimalizovaném tvaru (SOP). Na tuto funkci aplikujeme DeMorganovo pravidlo, pomocí kterého převedeme tvar s logickými součty na tvar s logickými součiny:

$$y = \overline{b} \cdot c + a \cdot \overline{b} \cdot \overline{c} => DeMorganovo\ pravidlo => \overline{\overline{\overline{bc} \cdot \overline{a\overline{b}\overline{c}}}}$$

Výslednou logickou funkci můžeme po této úpravě zapojit pouze pomocí obvodů typu NAND.

Realizace pomocí multiplexoru

U realizace obvodů pomocí tohoto způsobu je postup poněkud komplikovanější a jeho podrobný popis není předmětem této úlohy, Podrobný postup je uveden ve skriptech[1] na straně 91. U realizace pomocí této metody se vyhází ze základu funkce multiplexoru. Každý multiplexor obsahuje adresové, datové vstupy a vstup S – select, který povoluje, popřípadě zakazuje přepis hodnoty na datovém vstupu na výstup. Tohoto se využívá při skládání multiplexorů do větších funkčních celků. Počet datových vstupů závisí na počtu adresových vstupů: počet datových vstupů = 2^N, kde N je počet adresových vstupů. Adresové vstupy adresují příslušný datový vstup, který má být převeden na výstup. Tohoto se právě využívá u realizace kombinační logické funkce, kdy na datové vstupy připojíme v závislosti na příslušné funkce buď +5V nebo GND. Tímto zajistíme, že při příslušných vstupních proměnných, které jsou přivedeny na adresové vstupy, dostaneme na výstupu požadovanou logickou hodnotu.

Mohlo by se na první pohled zdát, že u multiplexoru s N adresovými vstupy můžeme vytvořit obvod pouze pro N vstupních proměnných, ale v praxi se využívá zapojení, ve kterých realizujeme pomocí multiplexoru s N adresovými vstupy logickou

funkci, která obsahuje N+1 vstupních proměnných. Postup návrhu takovýmto způsobem realizujeme následovně.

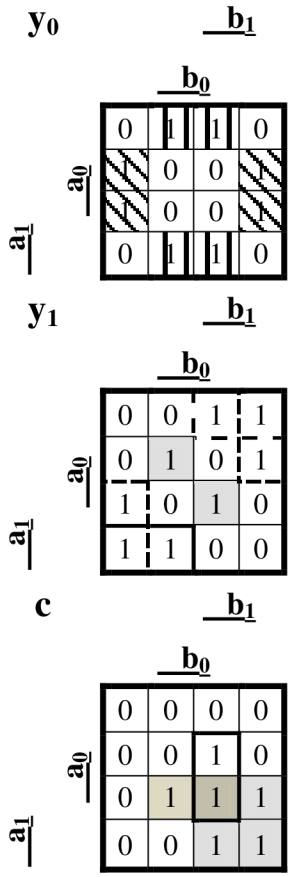
V prvním kroku výraz pro výslednou funkci doplníme tak, aby každý sčítanec obsahoval všechny eliminované proměnné. To znamená, že každý součinový člen vynásobíme závorkou, která obsahuje přímou i negovanou hodnotu chybějící eliminované proměnné. Ve druhém kroku závorky roznásobíme a vytkneme společné eliminované funkce. Ve třetím kroku všechny tyto funkce doplníme do tabulky, jejíž vstupní hodnoty budou eliminované funkce a výstupní hodnoty budou určeny zbytkovou funkcí.

SKLFPKO	BICT				Datum měření:						2013		Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

VYPRACOVÁNÍ – PRO UČITELE

1. Navrhněte pravdivostní tabulku pro dvoubitovou sčítačku s výstupním bitem pro přenos do vyššího řádu. Pomocí Karnaughovy mapy minimalizujte výsledné kombinačně logické funkce a zapojte je pomocí obvodu NAND.

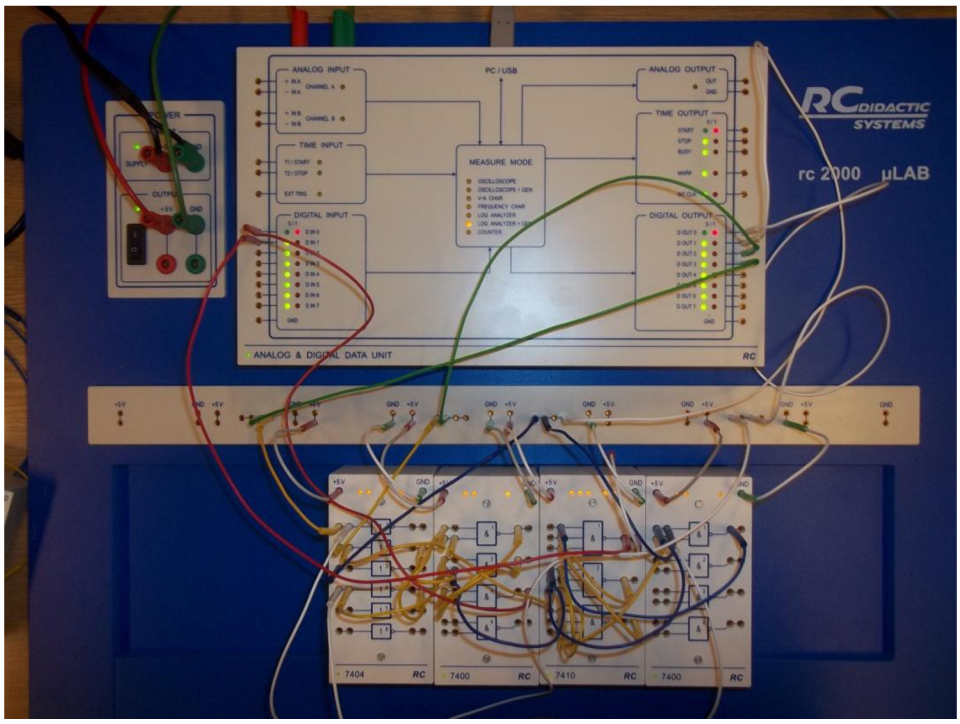
č.	a1	a0	b1	b0	c	y1	y0
0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	1	0
3	0	0	1	1	0	1	1
4	0	1	0	0	0	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	0	1	1
7	0	1	1	1	1	0	0
8	1	0	0	0	0	1	0
9	1	0	0	1	0	1	1
10	1	0	1	0	1	0	0
11	1	0	1	1	1	0	1
12	1	1	0	0	0	1	1
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	1
15	1	1	1	1	1	1	0



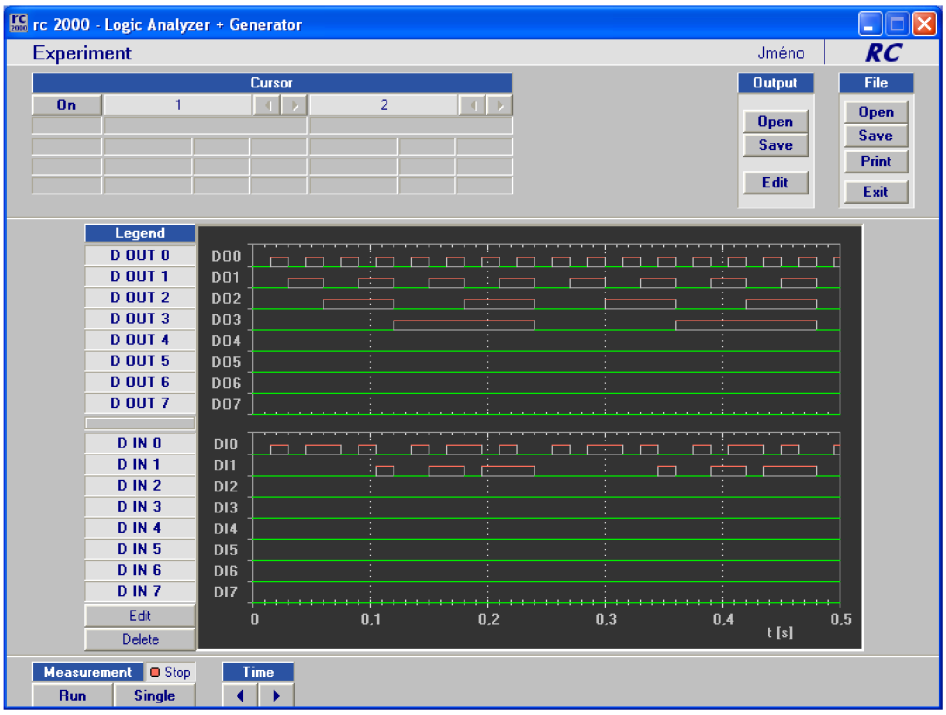
$$y_0 = \overline{b_0}a_0 + b_0\overline{a_0} = \overline{\overline{\overline{b_0}a_0} \cdot \overline{b_0\overline{a_0}}}$$

$$y_1 = b_1\overline{a_0a_1} + \overline{b_0}b_1\overline{a_1} + b_0\overline{b_1}a_0\overline{a_1} + \overline{b_0b_1}a_1 + \overline{b_1}\overline{a_0}a_1 + b_0b_1a_0a_1$$
$$= \overline{b_1\overline{a_0a_1}} \cdot \overline{\overline{b_0}b_1\overline{a_1}} \cdot \overline{b_0\overline{b_1}a_0\overline{a_1}} \cdot \overline{\overline{b_0b_1}a_1} \cdot \overline{\overline{b_1}\overline{a_0}a_1} \cdot \overline{b_0b_1a_0a_1}$$

$$c = b_1a_1 + b_0b_1a_0 + b_0a_0a_1 = \overline{\overline{b_1a_1} \cdot \overline{b_0b_1a_0} \cdot \overline{b_0a_0a_1}}$$



Obrázek 21 - zapojení úlohy 1

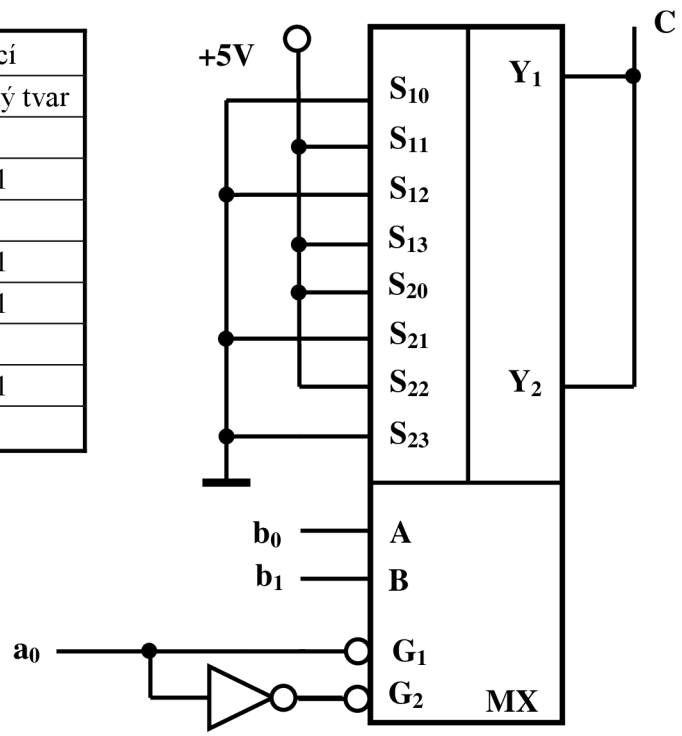


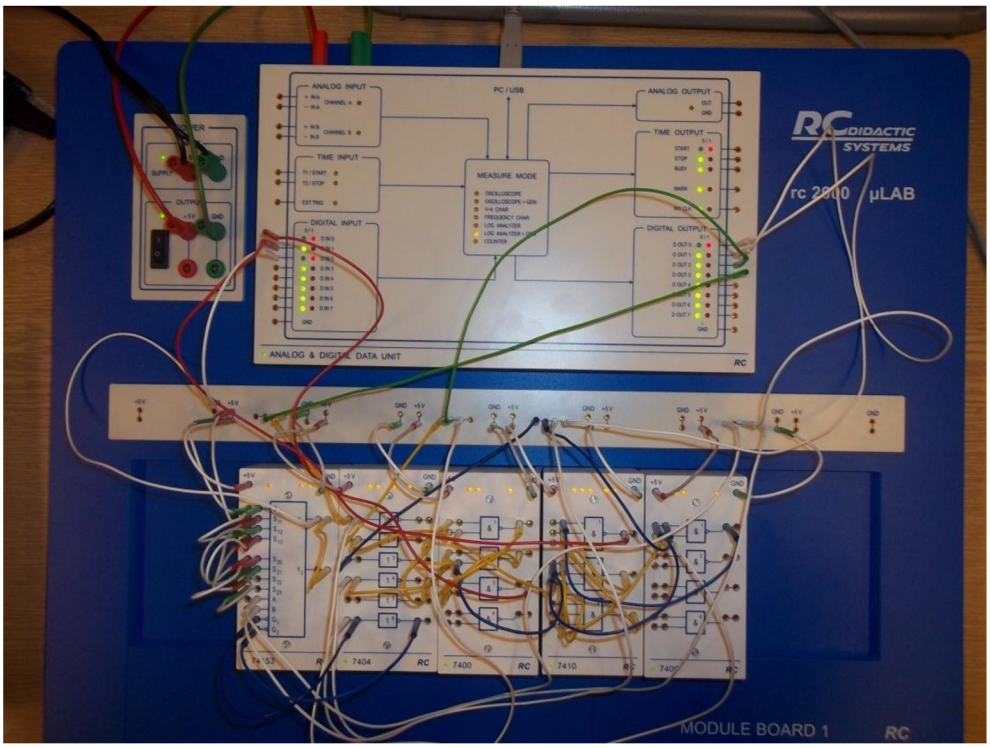
Obrázek 22 - časové průběhy výstupních funkcí D IN 0 = y_0, D IN 1 = C

2. Pomocí dvojného čtyřbitového multiplexoru realizujte funkci nižšího bitu sčítačky (y_0). Z důvodu kontroly vyučujícím, proveďte eliminaci pro bity b_0, b_1, a_0. Zapojení navrhněte tak, aby multiplexor pracoval jako osmibitový.

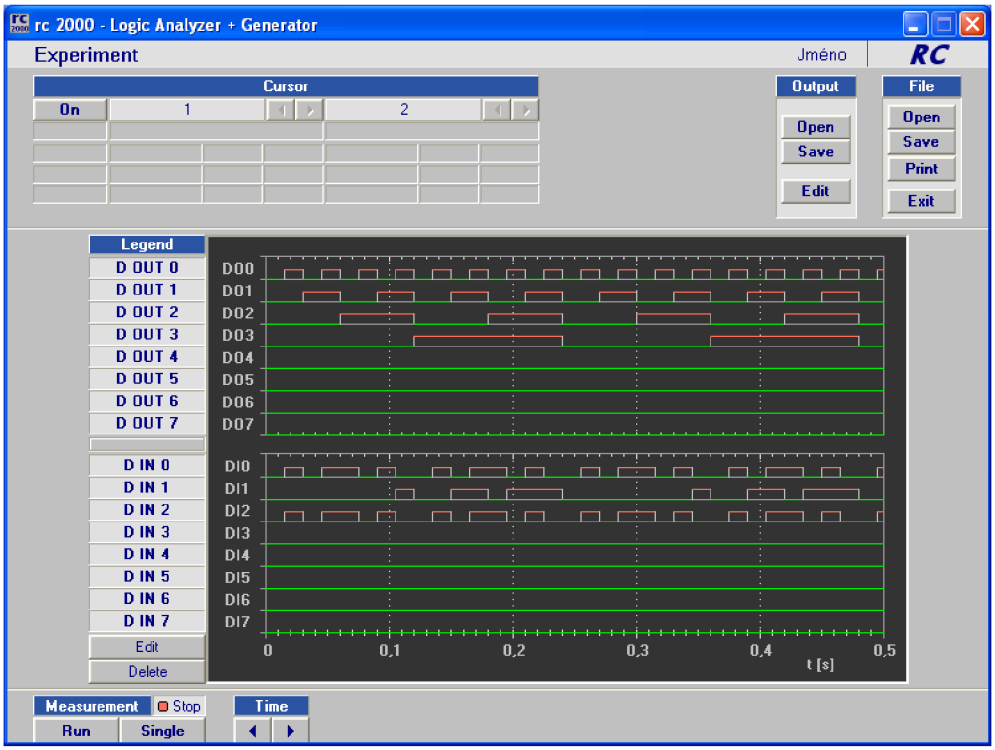
$$y_0 = b_0\overline{a_0} + \overline{b_0}a_0 = b_0\overline{a_0}(a_1 + \overline{a_1})(b_1 + \overline{b_1}) + \overline{b_0}a_0(a_1 + \overline{a_1})(b_1 + \overline{b_1})$$
$$= b_0b_1\overline{a_0}a_1 + b_0b_1\overline{a_0a_1} + b_0\overline{b_1a_0}a_1 + b_0\overline{b_1a_0a_1} + \overline{b_0}b_1a_0a_1$$
$$+ \overline{b_0}b_1a_0\overline{a_1} + \overline{b_0b_1}a_0a_1 + \overline{b_0b_1}a_0\overline{a_1}$$

Tabulka zbytkových funkcí				
b0	b1	a0	0-negace	1-přímý tvar
0	0	0	0	
0	0	1	a1+not(a1)=1	
0	1	0	0	
0	1	1	a1+not(a1)=1	
1	0	0	a1+not(a1)=1	
1	0	1	0	
1	1	0	a1+not(a1)=1	
1	1	1	0	



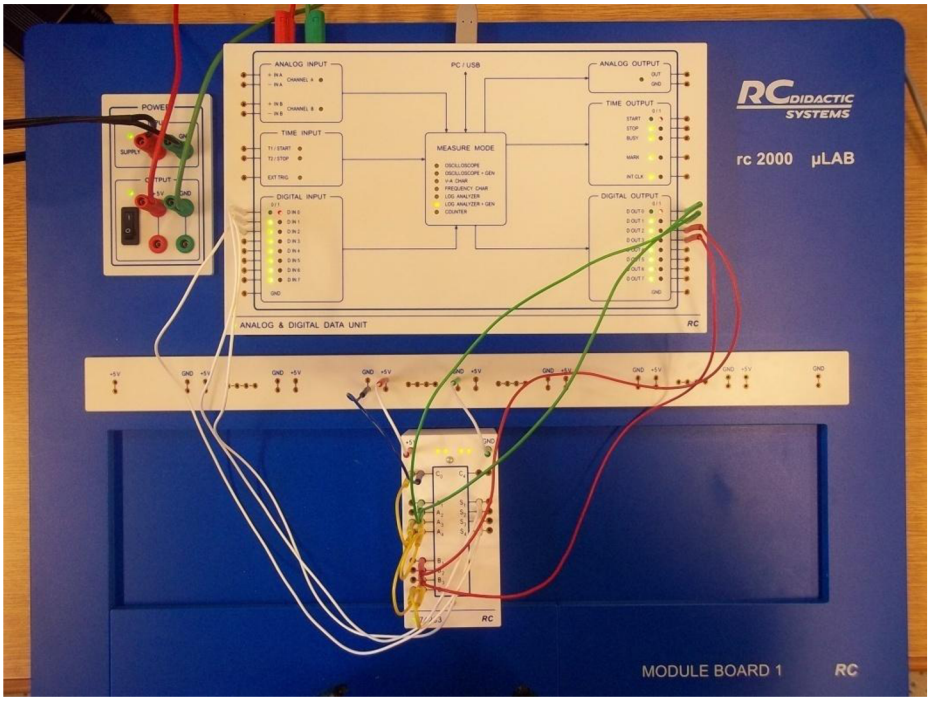


Obrázek 23 - zapojení úlohy 2

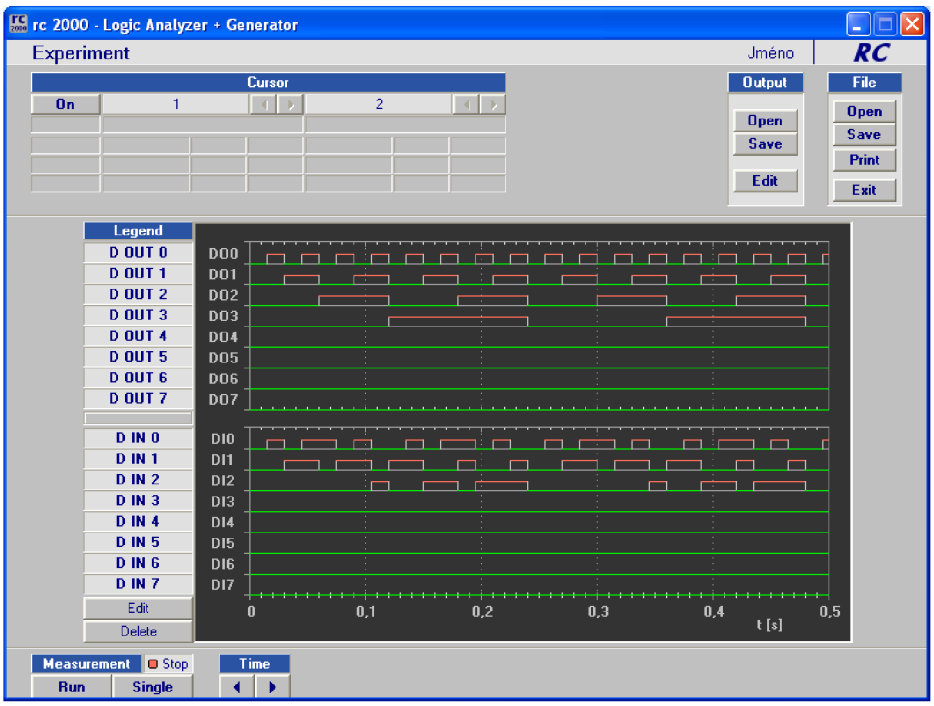


Obrázek 24 - časové průběhy výstupních funkcí D IN 0 = y_0, D IN 1 = C, D IN = y_0 pomocí MX

3. Ověřte funkci dvoubitové sčítačky pomocí integrovaného zapojení 74283.



Obrázek 25 - zapojení úlohy 3



Obrázek 26 - časové průběhy výstupních funkcí D IN 0 = y0, D IN 1 = y1, D IN 2 = C

A.3 Asynchronní, synchronní čítače a stavové automaty

ASČSA	BICT				Datum měření:							2013		Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

XX ASYNCHRONNÍ, SYNCHRONNÍ ČÍTAČE A STAVOVÉ AUTOMATY

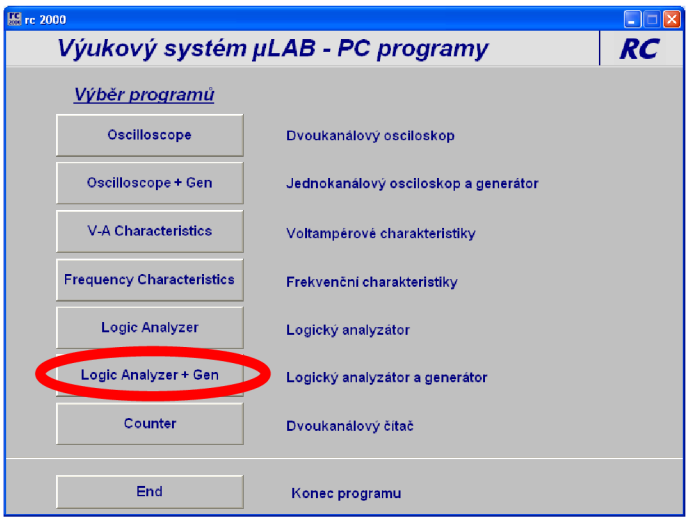
XX.1 Zadání

1. Navrhněte zapojení tříbitového asynchronního čítače vzad s neúplným cyklem 3,2,1,0,7,6 z obvodů typu JK (na nástupnou hranu). Obvod zapojte a ověřte jeho funkčnost připojením sedmi segmentového displeje a logického analyzátoru.
2. Navrhněte zapojení čtyřbitového synchronního čítače vzad s neúplným cyklem 9,8,7,6,5,4,3,2,1,0 kaskádně v připojeného k tříbitovému integrovanému čítači vzad 5,4,3,2,1,0 z obvodů typu 74193. Obvod zapojte a ověřte jeho funkčnost připojením sedmi segmentového displeje. Výsledný efekt bude odpočítávání jedné minuty.
3. Navrhněte zapojení tříbitového synchronního čítače vpřed na nástupnou hranu v binárním a Grayově kódu z obvodů typu D a ošetřete hazardy budící funkce, která bude mít jeden řídící vstup pro čítání v binárním (log 0) nebo Grayově (log 1) kódu. Obvod zapojte a ověřte jeho funkčnost sedmi segmentového displeje.

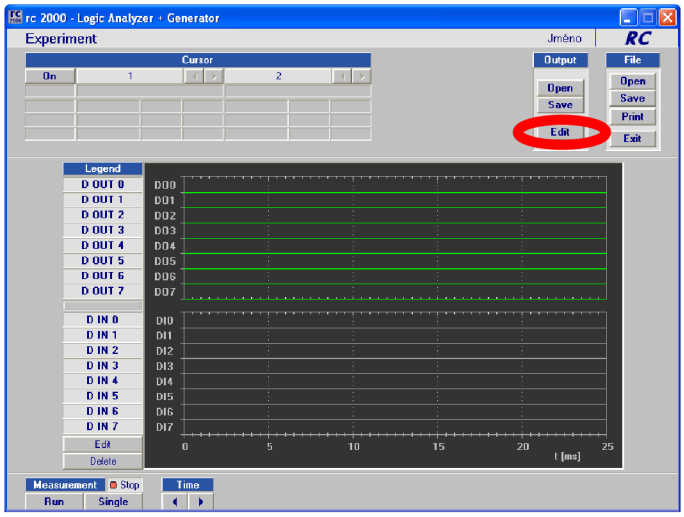
XX.2 Pokyny k zadání

1. Pomocí výukového systému RC Didactic zapojte návrh, který jste vypracovali v domácím úkolu. Pomocí obvodu 74112 realizujte zapojení, dávejte pozor na hodinový vstup tohoto obvodu, který je invertovaný. Připojte Analog and data unit k PC a spusťte program RC 2000, z nabídky vyberte Logic analyzer and Gen. Výstupy obvodů 74112 připojte na vstup logického analyzátoru (DIGITAL INP) a zároveň na blok se sedmi segmentovým displejem. V obslužném programu v bloku OUTPUT stiskněte EDIT a poté v bloku MODE stiskněte CLOCK. Periodu a časovou osu nastavte tak, aby perioda trvala 1s. Myší klikněte na D OUT 0 (zobrazí se průběh clk) a opusťte editor stisknutím LEAVE EDITOR. Digitální výstup připojte na hodinový vstup clk prvního obvodu. Stiskněte tlačítko RUN. Funkčnost obvodu nechte zkontrolovat vyučujícím.

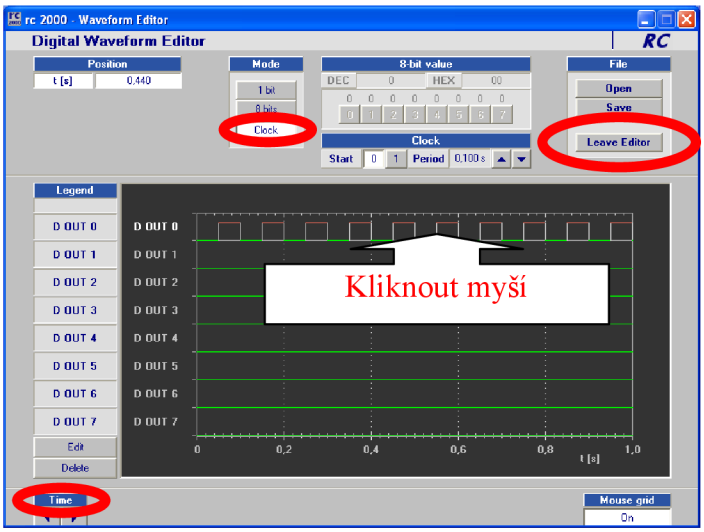
   Pozn.: Sedmi segmentový displej má čtyři vstupy, jelikož na něj bude připojen pouze tříbitový čítač, připojte nevyužitý vstup na log. 0.



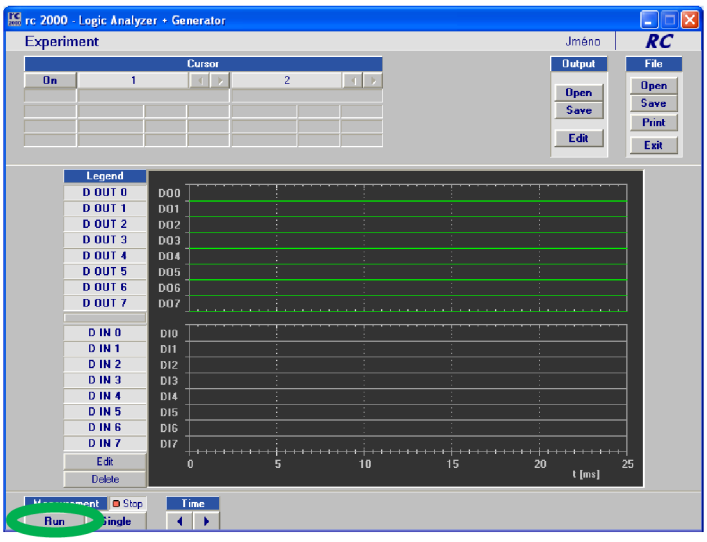
Obrázek 27 - výběr funkce logické jednotky



Obrázek 28 - výběr nastavení editoru výstupních signálů



Obrázek 29 - nastavení časových průběhů



Obrázek 30 - spuštění generátoru logických hodnot

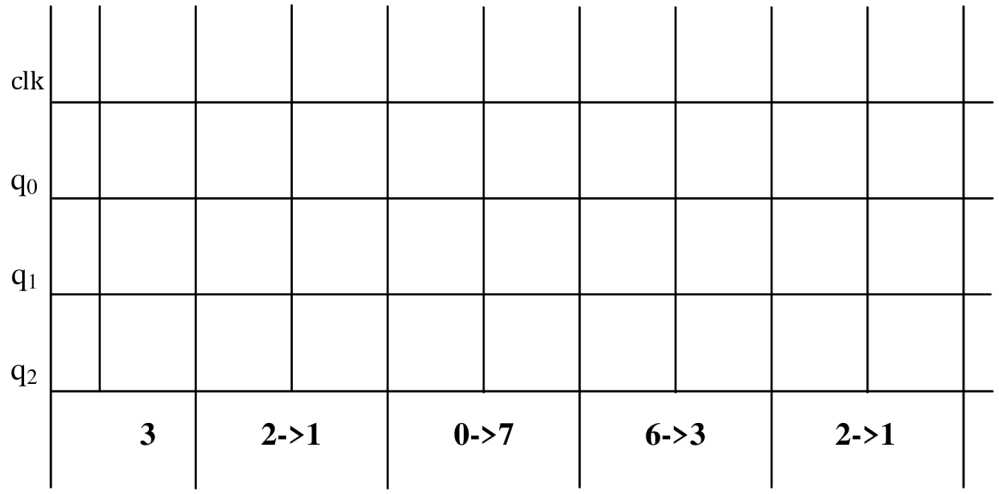
2. Pomocí integrovaného synchronního čítače 74193 zapojte obvod dle zadání, jako hodinový vstup využijte opět výstup digitální jednotky jako v předešlém bodě. Funkčnost zapojení nechte opět zkontrolovat vyučujícím.

3. Pomocí obvodu 7474 zapojte návrh binárního/Grayova čítače, který jste vypracovali v domácí přípravě. Využijte opět digitální jednotky. Funkci obvodu nechte zkontrolovat vyučujícím

ASČSA	BICT				Datum měření:						2013		Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

!!! Domácí příprava!!!

e) Nakreslete příslušné průběhy výstupů pro bod1 a nakreslete výsledné zapojení.



f) Doplňte pravdivostní tabulku pro bod 3. Příslušné funkce minimalizujte a ošetřete je proti hazardům.

č.	b/g	q1	q0	q1=d1	q0=d0	
0						BINARY
1						
2						
3						
4						GRAY
5						
6						
7						

XX. 3 Teoretické poznatky

Čítače tvoří velkou podskupinu číslicových sekvenčních systémů. Čítače můžeme rozdělit dle různých hledisek, z nichž nejčastější jsou:

- Podle kódu v němž pracují – BCD, Gray, binární, dekadické…
- Podle způsobu zapojení – synchronní, asynchronní
- Podle směru čítaní – vpřed, vzad
- Podle typu použitých obvodů v části registrů

Nejčastěji se setkáváme s čítači binárními a dekadickým. Tyto čítače se vyrábějí v nejrůznějších integrovaných variantách vybavené příslušnými řídícími vstupy a výstupy. Mezi nejznámější vstupy/výstupy patří SET - nastavení, RESET - nulování, CARRY - přetečení, BORROW - podtečení, ENABLE – povolení. Z hlediska konstrukce a vnitřního zapojení se budeme zajímat především o rozdělení na synchronní a asynchronní čítače.

Asynchronní čítače

Asynchronní čítače se vyznačují tím, že hodinové vstupy jednotlivých klopných obvodů (dále KLO) jsou zapojeny z výstupu předchozích klopných obvodů. Díky zpoždění jednotlivých klopných obvodů se může celkové zpoždění čítače nabývat větších hodnot než doba jedné periody vstupního kmitočtu. Jako základní zapojení můžeme uvažovat například čtyřbitový binární čítač vpřed z obvodu typu D. Z pravdivostní tabulky pro čítání vpřed je zřejmé, že změna vyššího bitu je podmíněná změnou bitu z 1 do 0 bitem nižším. Z tohoto vyplívá, že vstup CLK následujícího KLO je propojen s negovaným výstupem předchozího KLO. Na první pohled se může zdát, že asynchronní čítače jsou jednoduché na zapojení, ve skutečnosti již malá změna v čítacím pořadí popřípadě zkrácení cyklu vyžaduje složitější uvažování a analýzu obvodu. Pro zapojení čítače z klopných obvodů JK se postupuje pomocí tabulek pro klopný obvod JK. Podrobný popis naleznete ve skriptech[1].

Synchronní čítače, stavové automaty

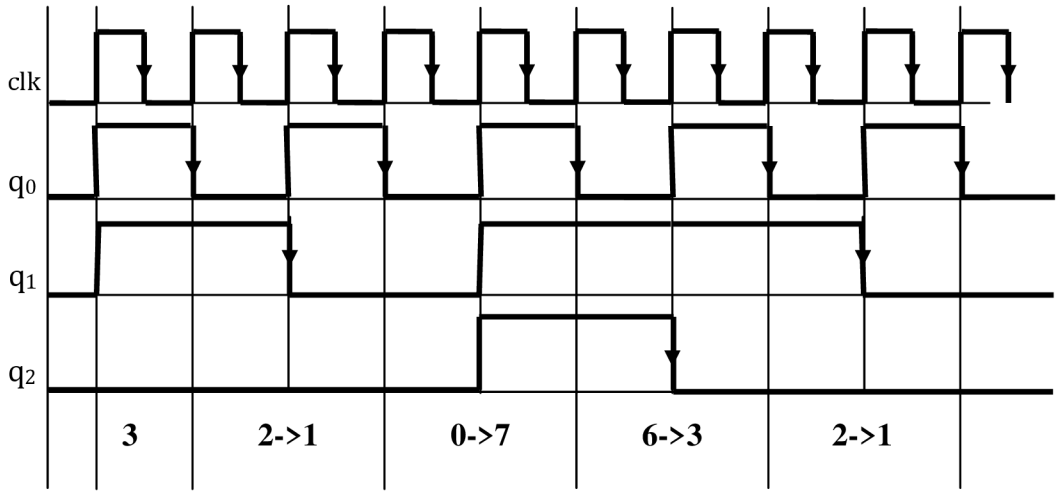
Synchronní čítače a stavové automaty se vyznačují především tím, že jejich hodinové vstupy jsou zapojeny paralelně na řídící hodinový signál. Dále tyto obvody obsahují kombinační část, která v podstatě určuje posloupnost stavů na výstupu, proto můžeme čítače i stavové automaty zahrnout do jedné skupiny. Jelikož jsou všechny registry řízeny ve stejném okamžiku nástupnou nebo sestupnou hranou hodinového signálu, případné hazardní stavy v kombinační části mají mezi dvěma následujícími sestupnými hranami dostatek času, aby se ustálily. Návrh synchronního čítače nebo stavového automatu se dá jednoduše algoritmizovat. Pravdivostní tabulka návrhu se skládá v podstatě ze tří částí. V první části tabulky vypíšeme postupně posloupnost čítání. Ve druhé části vypíšeme do tabulky následující stavy čítání vzhledem k první části tabulky. Ve třetí části zapíšeme budící funkci. U obvodu typu D je budící funkce rovna následujícímu stavu, takže se výsledná tabulka skládá v podstatě ze dvou částí. U obvodu typu JK vždy vycházíme z tabulky změn stavů výstupu obvodu v závislosti na hodnotách J a K.

ASČSA	BICT				Datum měření:						2013		Příjmení a jméno:
					Den (vyznačte X):				Po	Út	St	Čt	Pá	
Hodina (vyznačte X):	7	8	9	10	11	12	13	14	15	16	17	18	19	

VYPRACOVÁNÍ – PRO UČITELE

1. Navrhněte zapojení tříbitového asynchronního čítače vzad s neúplným cyklem 3,2,1,0,7,6 z obvodů typu JK (na nástupnou hranu). Obvod zapojte a ověřte jeho funkčnost připojením sedmi segmentového displeje a logického analyzátoru.

Řešení:



q_0 – mění se při sestupné hraně CLK => Zapojení jako dělička dvěma => J K = "1"

q_1 – mění se při nástupných hranách q_0, je-li q_2 = "1" ke změně nedojde => J,K prvního klopného obvodu připojíme $\overline{q_2}$ na CLK přivedu $\overline{q_0}$

q_2 – na CLK $\overline{q_0}$, J K viz tabulka 0. buňka – odpovídá změně z 0 -> 7, 1. Buňka - odpovídá změně 2->1, 3. buňka – volná, 4. Buňka - odpovídá změně 6-> 3. Hodnoty viz tabulka následujících a minulých funkcí obvodu J K

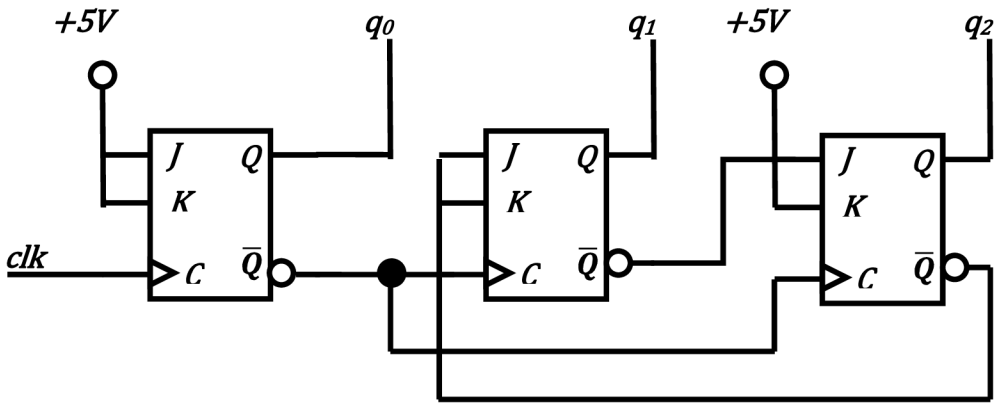
K_2

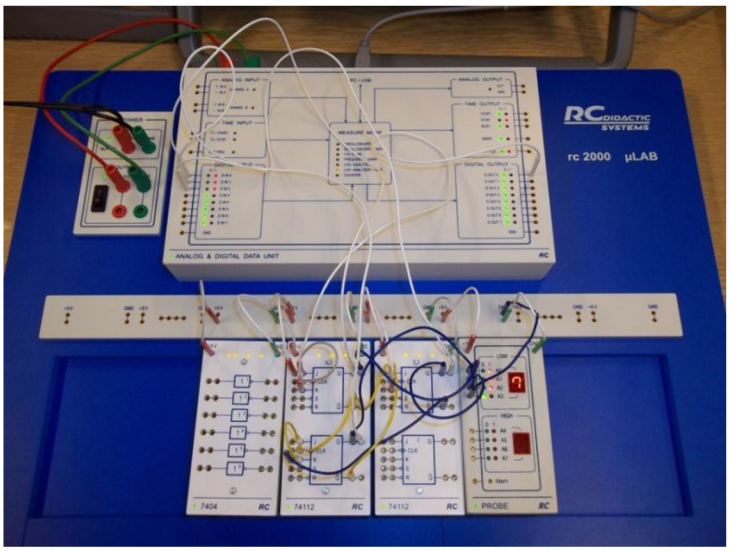
q_2 \ q_1		
	x	x
	x	1

J_2

q_2 \ q_1		
	1	0
	x	x

Zapojení:

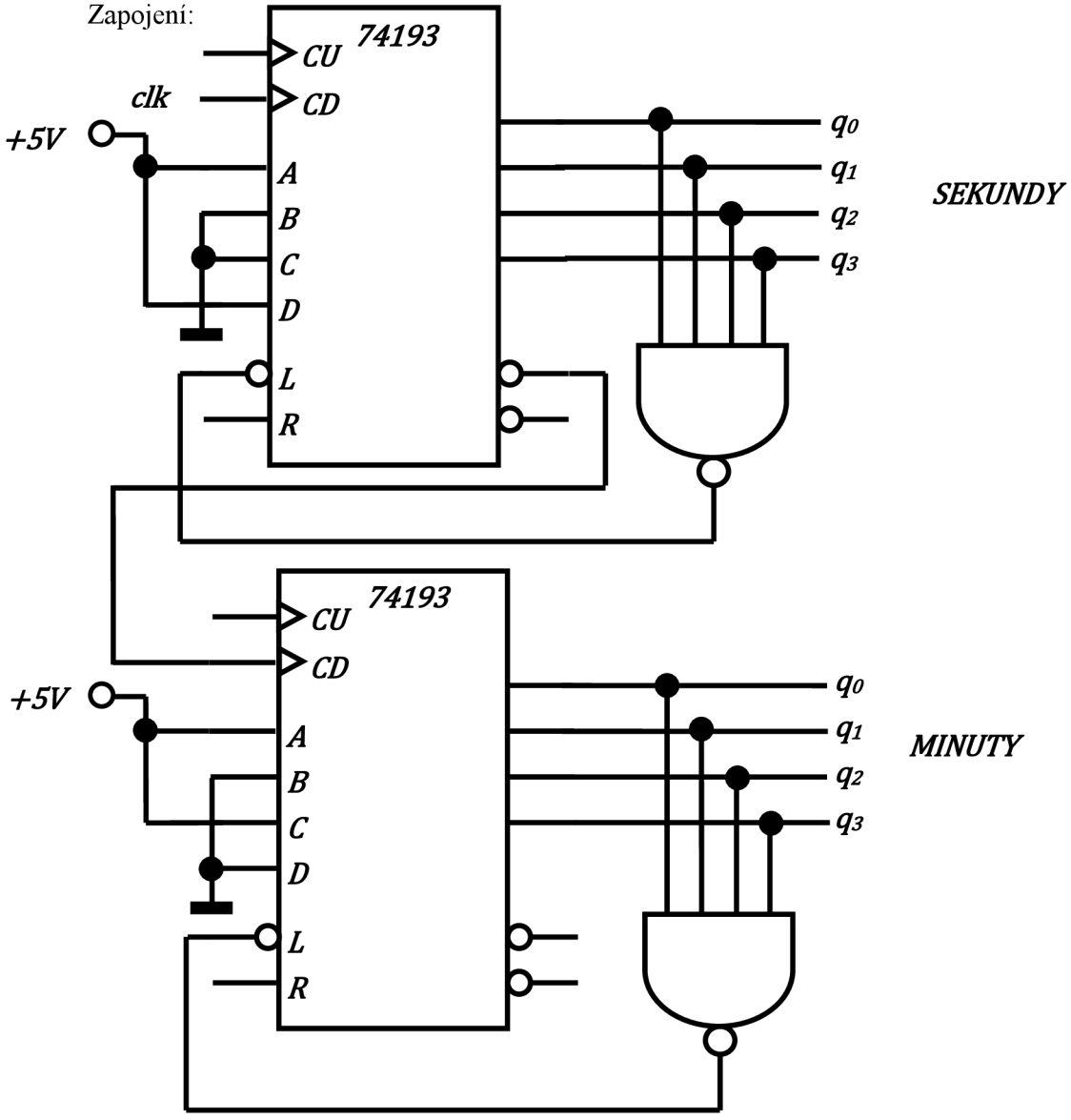


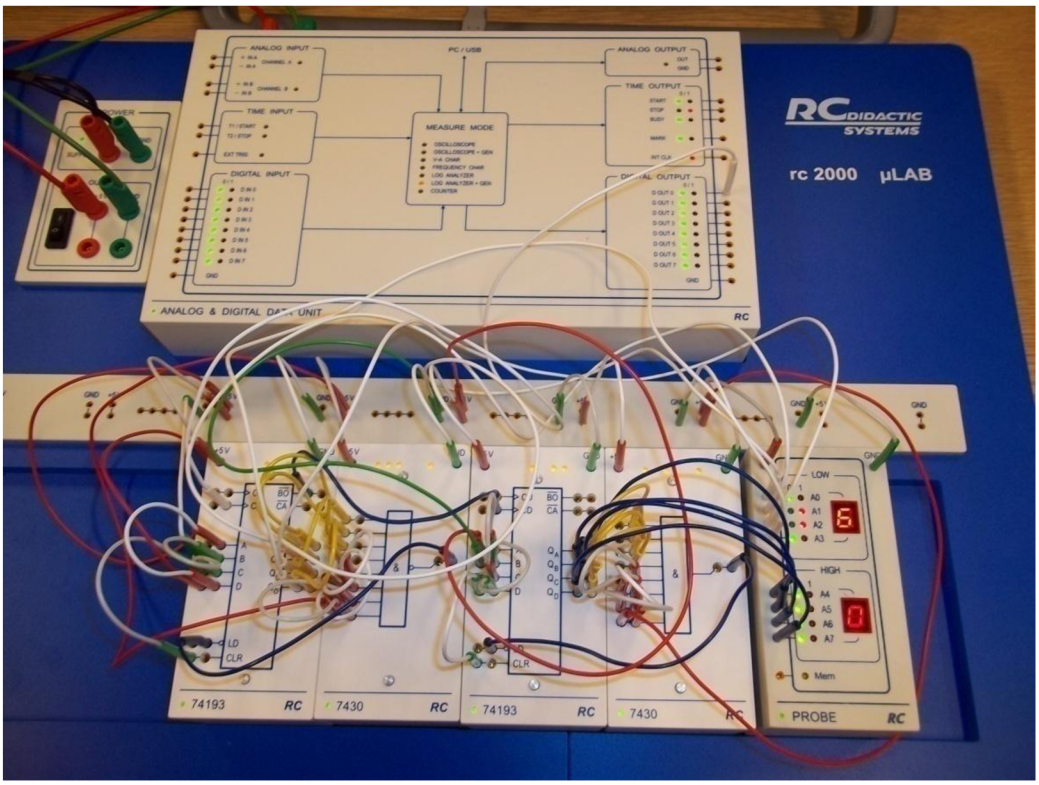


Obrázek 31 - zapojení 1. úkolu

2. Navrhněte zapojení čtyřbitového synchronního čítače vzad s neúplným cyklem 9,8,7,6,5,4,3,2,1,0 kaskádně v připojeného k tříbitovému integrovanému čítači vzad 5,4,3,2,1,0 z obvodů typu 74193. Obvod zapojte a ověřte jeho funkčnost připojením.

Zapojení:





Obrázek 32 - zapojení 2. úkolu

3. Navrhněte zapojení tříbitového synchronního čítače vpřed na nástupnou hranu v binárním a Grayově kódu z obvodů typu D a ošetřete hazardy budící funkce, která bude mít jeden řídící vstup pro čítání v binárním (log 0) nebo Grayově (log 1) kódu. Obvod zapojte a ověřte jeho funkčnost sedmi segmentového displeje.

č.	b/g	q1	q0	q1=d1	q0=d0	
0	0	0	0	0	1	**BINARY**
1	0	0	1	1	0	
2	0	1	0	1	1	
3	0	1	1	0	0	
4	1	0	0	0	1	**GRAY**
5	1	0	1	1	1	
6	1	1	0	1	0	
7	1	1	1	0	0	

D_0

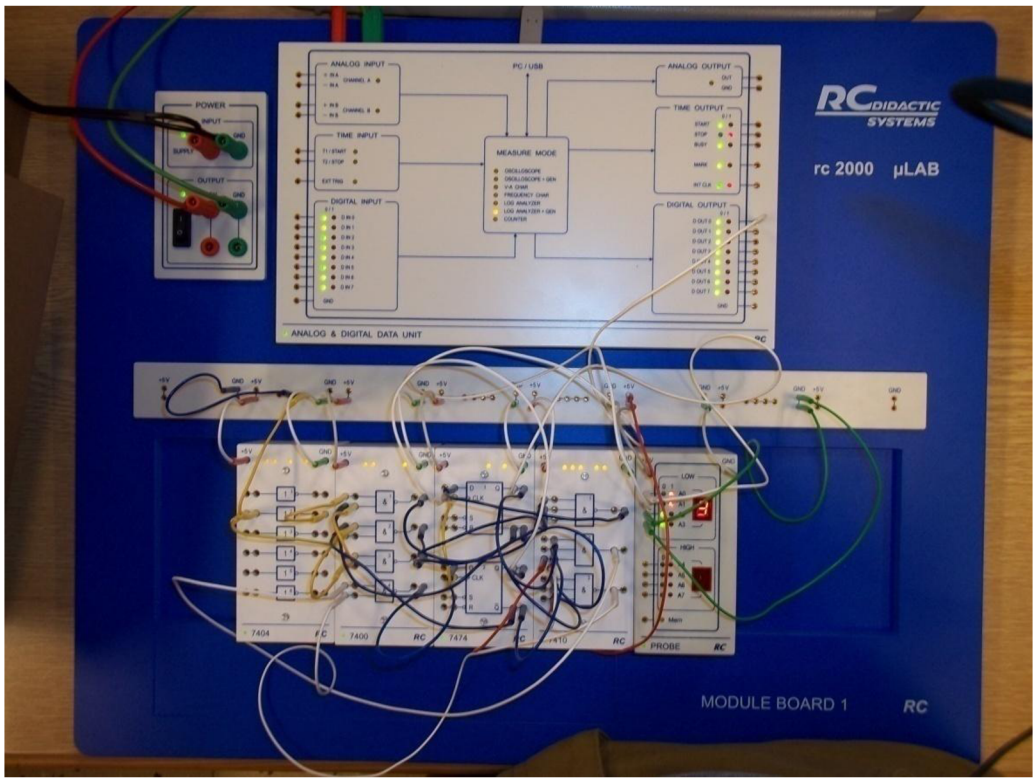
b/g \ q_1q_0	00	01	11	10
0	1	0	0	1
1	1	1	0	0

D_1

b/g \ q_1q_0	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$$d_0 = \overline{q_0} \cdot \overline{bg} + \overline{q_1} \cdot bg + \overline{q_0} \cdot \overline{q_1} = \overline{\overline{\overline{q_0} \cdot \overline{bg}} \cdot \overline{\overline{q_1} \cdot bg} \cdot \overline{\overline{q_0} \cdot \overline{q_1}}}$$

$$d_1 = q_0 \cdot \overline{q_1} + \overline{q_0} \cdot q_1 = \overline{\overline{q_0 \cdot \overline{q_1}} \cdot \overline{\overline{q_0} \cdot q_1}}$$



Obrázek 33 - zapojení 3. úkolu

B JAVA APPLETY

B.1 74138 APPLET

Applet přiložen na CD v souboru */Java/74138

B.2 74151 APPLET

Applet přiložen na CD v souboru */Java/74151

B.3 74688 APPLET

Applet přiložen na CD v v souboru */Java/74688

B.4 JK Synchronous Counter APPLET

Applet přiložen na CD v v souboru */Java/JKSynchronousBinaryCounter

B.5 Karnaugh map APPLET

Applet přiložen na CD v souboru */Java/Karnaugh Map

C SIMULACE V PROGRAMU ISE XILINX

C.1 Simulace obvodu 74138

Projekt simulace obvodu 74138 přiložen na CD v souboru */ISE/z74138

C.2 Simulace obvodu 74151

Projekt simulace obvodu 74151 přiložen na CD v souboru */ISE/mpx74151

C.3 Simulace obvodu 74688

Projekt simulace obvodu 74688 přiložen na CD v souboru */ISE/com74688

C.4 Simulace obvodu synchronního čítače z JK

Projekt simulace obvodu synchronního čítače z JK přiložen na CD v souboru */ISE/SC