VYBRANÉ PŘÍKLADY ZE ZKOUŠEK

VERZE SEZNAMU 2015-V1.1 BUILD 2016.01.06.2000

- vybrané příklady pocházejí z předchozích zkoušek a slouží k ilustraci, co můžete očekávat v písemce;
- nejde o úplný seznam, u zkoušky se mohou objevit i jiné příklady, a samozřejmě vždy budou jiné vstupní hodnoty;
- seznam rovněž neobsahuje teoretické otázky, tedy otázky, na které se očekává slovní odpověď;
- u příkladů, kde není uvedeno řešení, si můžete svoji ideu ověřit simulací či v přednáškách;
- označování příkladů má pouze pojmenovávací účel (jde o interní reference pro potřeby vyučujícího), takže kódy se rozhodně nevztahují ke složitosti či k typu úlohy.
- příklady mohou pochopitelně obsahovat i nechtěné překlepy - na ty ale nutno upozornit před zkouškou, u ní bude už pozdě.

B1. Rozepište funkci napsanou ve VHDL syntaxi tak, aby obsahovala jen operace "or", "and" a "not", přičemž operace "not" nebyla nikde aplikovaná na výraz v závorce. A, B, C, F1 a F2 jsou typu boolean.

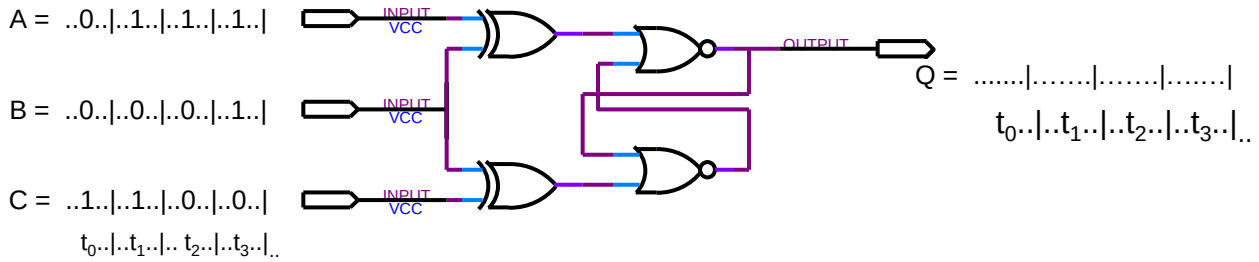
F1<= A xor (B or C);

F1<=.....

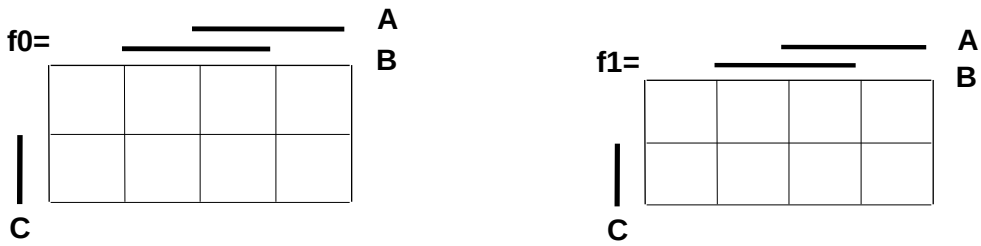
F2<= **not** (A and (B **xor** C));

F2<=.....

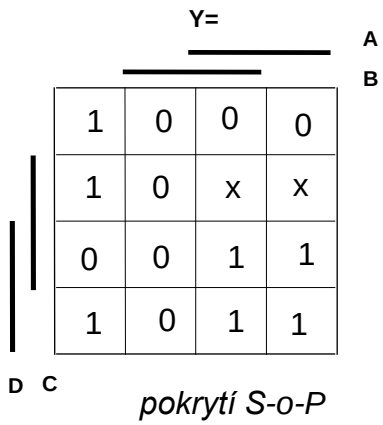
B3. Vstupy A, B, C měly v časech t_0, t_1, t_2, t_3 hodnoty uvedené v obrázku. Napište hodnoty výstupu Q. Předpokládejte, že intervaly mezi změnami vstupů jsou tak dlouhé, že lze zanedbat zpoždění hradel.



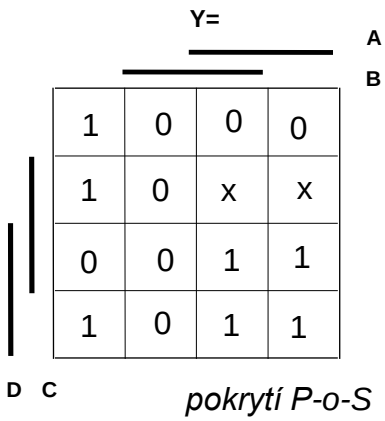
B4. Funkci Q=f(A,B,C,Q) z otázky B3 rozložte na tvar $Q=\overline{Q}.f0(A,B,C) + Q.f1(A,B,C)$ pomocí Shannonovy dekompozice. Výsledné funkce f0 a f1 napište jako Karnaughovy mapy



B5. V Karnaughově mapě vyznačte minimální pokrytí S-o-P (Sum-of-Products) a P-o-S (Product-of-Sums). Volte jen taková pokrytí, která nevytvářejí hazardy.



pokrytí S-o-P



pokrytí P-o-S

A1. K negované funkci $\overline{F_x}$ napište její nenegovanou analogii F_x

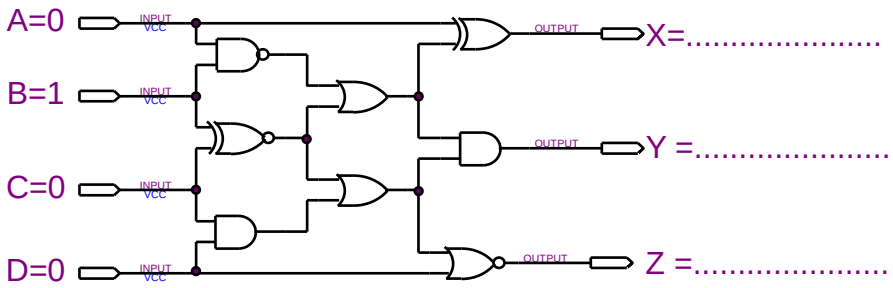
$\overline{F_1}=\overline{A}.B.\overline{C} + A.\overline{B}.C + \overline{A}.B + A.\overline{C} + \overline{B}.\overline{C}$

F_1=.....

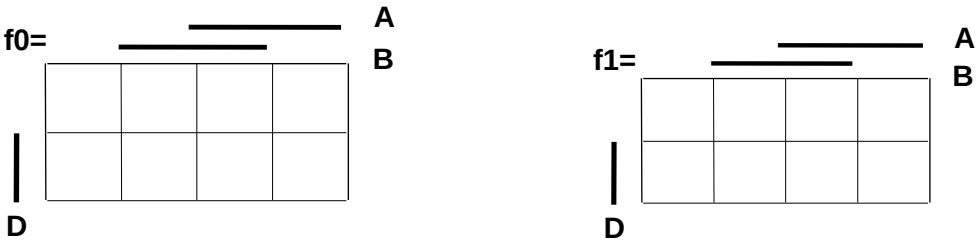
$\overline{F_2}=(\overline{A}+B+\overline{C})\ .(A+\overline{B}+C)\ .(\ \overline{A}+B).\ (A+\overline{C})\ .(\overline{B}+\overline{C})$

F_2=.....

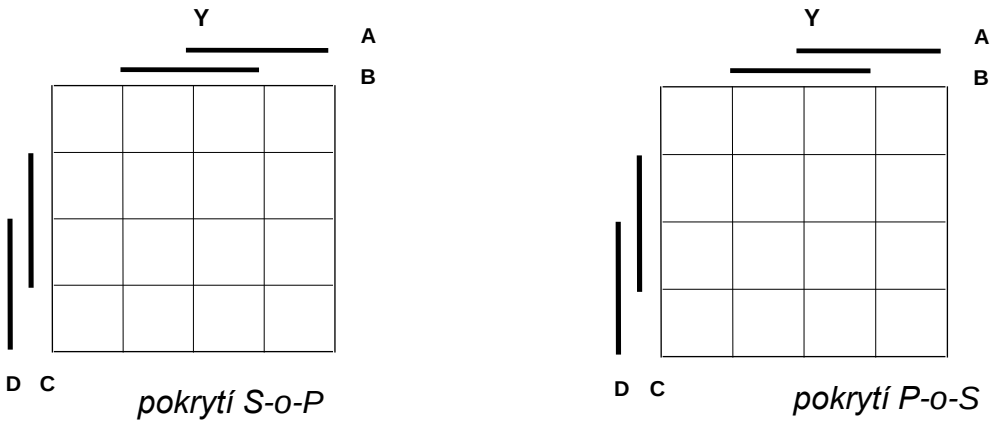
A3. Funkce má vstupní hodnoty uvedené v obrázku, napište výstupní hodnoty (3 body)



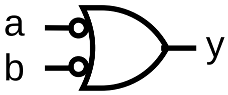
A4. Funkci Y z otázky A3 rozložte na tvar Y=f(A,B,C,D)=$\overline{C}$.f0(A,B,D) + C.f1(A,B,D) pomocí Shannonovy dekompozice. Výsledné funkce f0 a f1 napište jako Karnaughovy mapy (4 body)



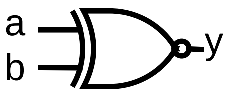
A5. Funkci Y z otázky 3 napište jako Karnaughovu mapu a vyznačte v ní minimální pokrytí S-o-P (Sum-of-Products) a P-o-S (Product-of-Sums). Volte taková pokrytí, která nevytvářejí hazardy.



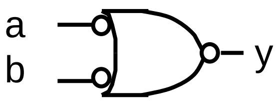
C1. Ve schématu vidíte následující hradla. Napište jejich pravdivostní tabulky.



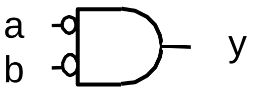
a	b	Y
0	0	
0	1	
1	0	
1	1	



a	b	Y
0	0	
0	1	
1	0	
1	1	

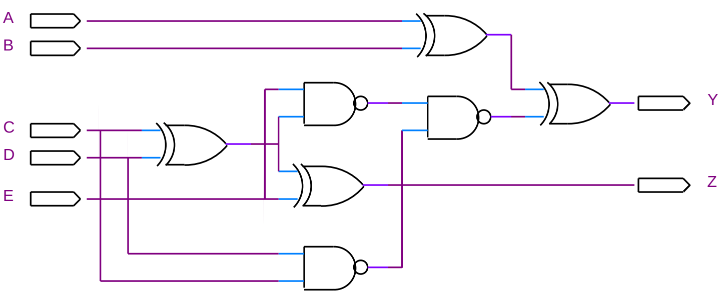


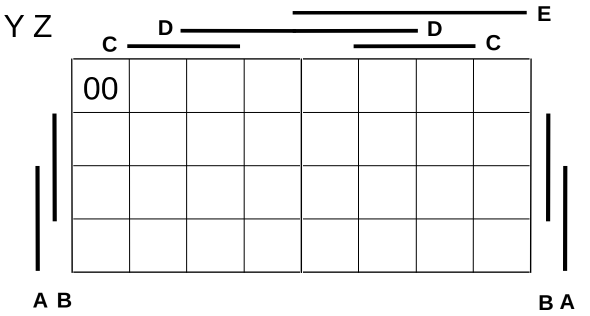
a	b	Y
0	0	
0	1	
1	0	
1	1	



a	b	Y
0	0	
0	1	
1	0	
1	1	

D1. *Ve schématu vidíte následující zapojení. Napište jeho pravdivostní tabulku. Výstupy YZ pište do společné tabulky jako dvojice bitů v pořadí Y Z.*



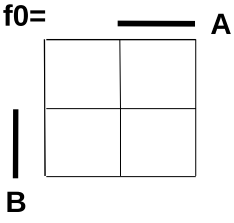


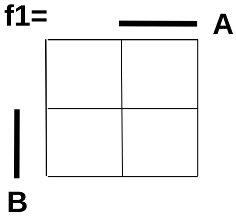
D2. *Obrázku obvodu z otázky 1 chybí titulek. Správně pojmenujte jeho funkci s použitím nejvýše 7 slov (pozn. nepočítají se jednopísmenné spojky a předložky a jména signálů z obrázku).*

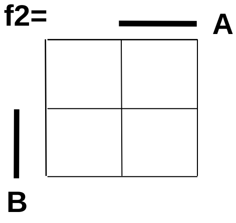
....

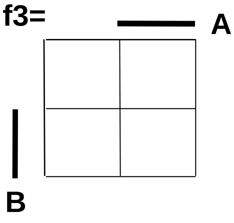
D3. *Funkci* Q:=f(A,B,C,D) := (A or B) and (((A and not B) xor (D and not A)) or (C and not D));

rozložte na tvar Q=$\overline{C}$.$\overline{D}$.f0(A,B) + $\overline{C}$.D.f1(A,B) + C.$\overline{D}$.f2(A,B) + C.D.f3(A,B) *pomocí Shannonovy dekompozice. Výsledné funkce f0, f1, f2 a F3 napište jako Karnaughovy mapy*:

f0=


f1=


f2=


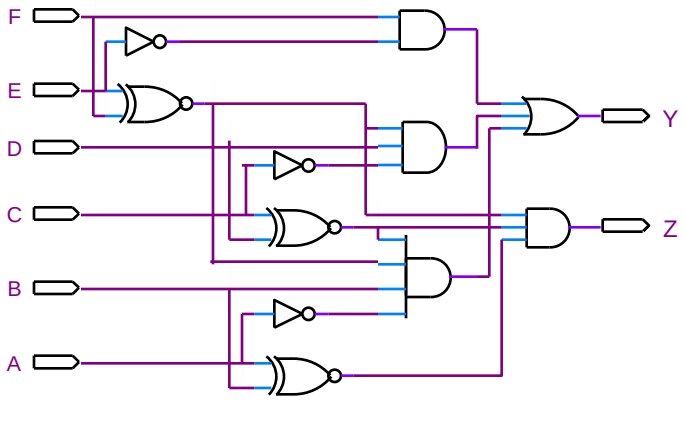
f3=


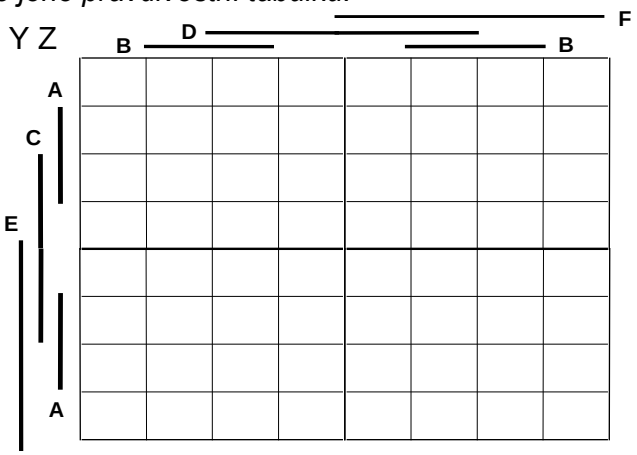
D4. *Zaškrtnutím označte všechny logické funkce, které zde mají jinou funkci s nimi shodnou:*

f1<=(A xor C) or (A and not C);
f2<=(B or C) and (not A or B or C);
f3<=((C and not B) or (B and A));
f4<=(A or C) and (not A or not C);
f5<=(A and not B) xor (A and C);
f6<=(A and not C) or (C and not A);

f1 ☐
f2 ☐
f3 ☐
f4 ☐
f5 ☐
f6 ☐

E1. *Ve schématu vidíte následující zapojení. Napište jeho pravdivostní tabulku.*



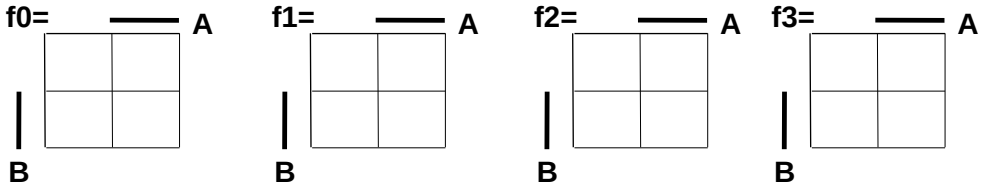


E2. *Připojením hradla **nor** k výstupům Y a Z obvodu z otázky E1, vznikne další výstup **X:=not(Y or Z)**. Pojmenujte operaci prováděnou novou funkcí **X:=f(A,B,C,D,E,F)**, a to s použitím nejvýše 6 slov:*

.....

E3. *Funkci* Q:=f(A,B,C,D) := (A or B) and (((A and not B) xor (D and not A)) or (C and not D));

rozložte na tvar $Q=\overline{C}.\overline{D}.f0(A,B) + \overline{C}.D.f1(A,B) + C.\overline{D}.f2(A,B) + C.D.f3(A,B)$ *pomocí Shannonovy dekompozice. Výsledné funkce f0, f1, f2 a F3 napište jako Karnaughovy mapy*:



E4. *Zaškrtnutím označte všechny logické funkce, které zde mají jinou funkci s nimi shodnou:*

f1<=((C and not B) or (C and B and A));	f1 ☐
f2<=(A xor C) or (A and not C);	f2 ☐
f3<=(A or B) and (not A or B or C);	f3 ☐
f4<=(not A or not C) and (C or A);	f4 ☐
f5<=(not C and A) or (not A and C);	f5 ☐
f6<=(A and not B) xor (A and and not B and C);	f6 ☐

E4x- Varianta E4 z přednášek – jsou uvedené funkce stejné?

x1<=(A and not C) or (C and not A); *x2<=(A and not B) xor (A and C);*

x3<=(A or C) and (not A or not C); *x4<=(A xor C) or (A and not C);*

C4. Zaškrtnutím označte všechny logické funkce, které zde mají jinou funkci s nimi shodnou:

f1<=(((not B and not A) or (A and D)) and C) or (B and A);	f1 ☐
f2<=((((not A or D) and C) or A) and B);	f2 ☐
f3<=((C and (not A or D)) or B) and A and not B;	f3 ☐
f4<=((not A or D) and C and not B) or (B and A);	f4 ☐
f5<=(A and not B) or (A and D) or (B and A);	f5 ☐
f6<=(B or C) and (not A or B or D) and (A or not B);	f6 ☐

Z technických důvodů - umísťování obrázků na stránce
- nejdou všechny odpovědi nutně za sebou ve stejném pořadí jako zadané příklady

B1.

F1<= A xor (B or C);

F1<=(A and not B and not C) or not A and (B or C);……

F2<= **not** (A and (B **xor** C));

F2<=not A or ((not B or C) and (B or not C));……

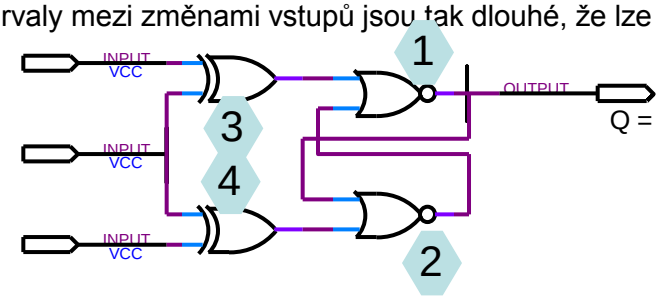
B3. Vstupy A, B, C měly v časech t_0, t_1, t_2, t_3 hodnoty uvedené v obrázku. Napište hodnoty výstupu Q. Předpokládejte, že intervaly mezi změnami vstupů jsou tak dlouhé, že lze zanedbat zpoždění hradel.

A = ..0..|..1..|..1..|..1..|

B = ..0..|..0..|..0..|..1..|

C = ..1..|..1..|..0..|..0..|

t_0..|..t_1..|.. t_2..|..t_3..|..



Q = ...1...|…0….|…0….|…1….|

t_0..|..t_1..|.. t_2..|..t_3..|..

B4. Funkci Q=f(A,B,C,Q) z otázky B3 rozložte na tvar $Q=\overline{Q}.f0(A,B,C) + Q.f1(A,B,C)$ pomocí Shannonovy dekompozice. Výsledné funkce f0 a f1 napište jako Karnaughovy mapy.

Řešení: Schéma je v podstatě grafickým zápisem logického výrazu. Použije postup ukazovaný na 2. přednášce. Očíslujeme-li hradla a postupně přepíšeme jejich operace do výrazu:

$$Q := \overline{1} := \overline{3} + \overline{2} := \overline{(A \text{ xor } B) + \overline{(Q + \overline{4})}} := \overline{(A \text{ xor } B) + \overline{(Q + (B \text{ xor } C))}}$$

Pro zjednodušení rozepíšeme horní negaci podle de Morganova pravidla (předn. 2) Využijeme toho, že operace xor je vlastně non-ekvivalence, a tak platí (A xor B) je (A ≠ B), a $\overline{(A \text{ xor } B)}$ je (A ≡ B)

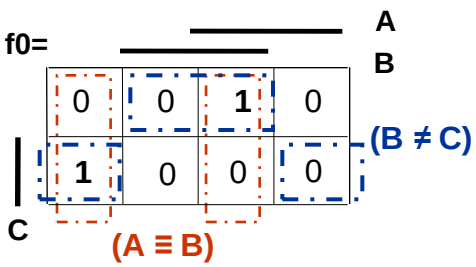
$$Q := \overline{(A \text{ xor } B)} . (Q + (B \text{ xor } C)) := \mathbf{(A \equiv B) . (Q + (B \neq C))}$$

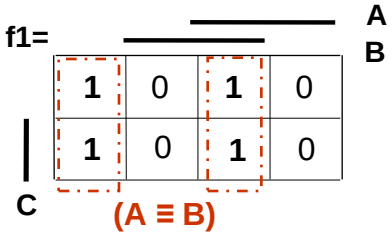
Z posledního tvaru již snadno spočítáme koofaktory f0 a f1 Shannonovy dekompozice (přednáška 4) tím, že zjistíme hodnoty funkce f(A,B,C,Q) v bodech Q='0' a Q='1', a to dosazením za Q.

f0 := f(A,B,C,'0') := (A ≡ B) . ('0' + (B ≠ C)) := (A ≡ B) . ('0' + (B ≠ C)):= **(A ≡ B) . (B ≠ C)**

f1 := f(A,B,C,'1') := (A ≡ B) . ('1' + (B ≠ C)) := (A ≡ B) . '1' := **(A ≡ B)**

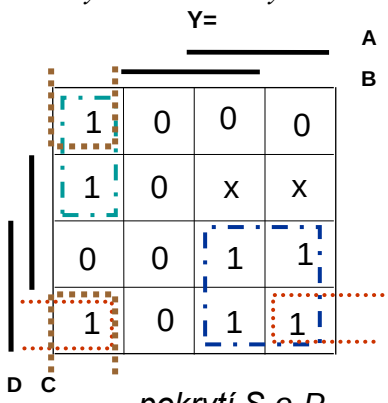
*Obě nalezené funkce f0:=***(A ≡ B) . (B ≠ C)** *a f1:=***(A ≡ B)** *zapíšeme jako Karnoughovy mapy*



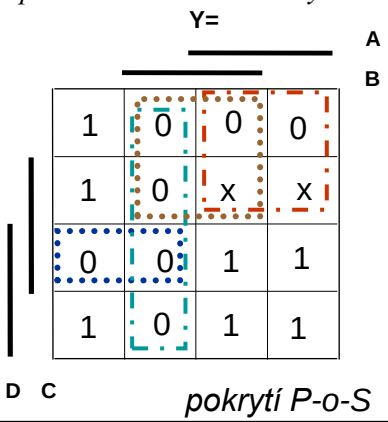


B5. V Karnaughově mapě vyznačte minimální pokrytí S-o-P (Sum-of-Products) a P-o-S (Product-of-Sums). Volte jen taková pokrytí, která nevytvářejí hazardy.

Řešení - Podle přednášky o hazardech vytvoříme souvislé pokrytí, a to přidáním hnědě označených členů.



pokrytí S-o-P



pokrytí P-o-S

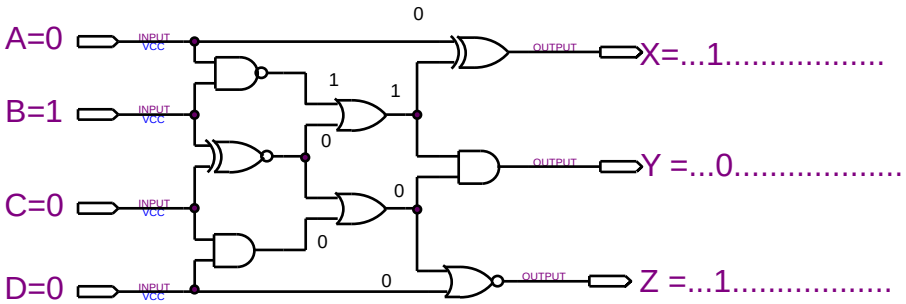
A1. $\overline{F_1}=\overline{A}.B.\overline{C} + A.\overline{B}.C + \overline{A}.B + A.\overline{C} + \overline{B}.\overline{C}$

$F_1= (A+\overline{B}+C) . (\overline{A}+B+\overline{C}) . (A+\overline{B}) . (\overline{A}+C) .(B.+C)$……

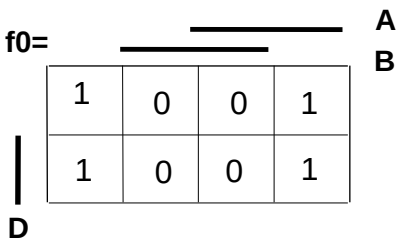
$\overline{F_2}=(\overline{A}+B+\overline{C}) .(A+\overline{B}+C) .(\overline{A}+B). (A+\overline{C}) .(\overline{B}+\overline{C})$

$F_2= A.\overline{B}.C + \overline{A}.B.\overline{C} + A.\overline{B} + \overline{A}.C + B.C$……

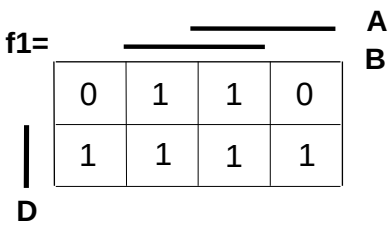
A3



A4

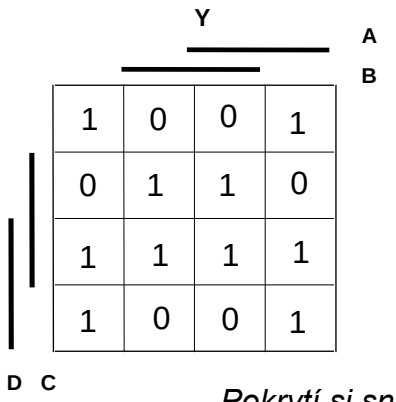


1	0	0	1
1	0	0	1



0	1	1	0
1	1	1	1

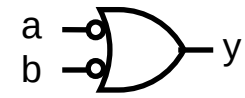
A5 neporytá KM



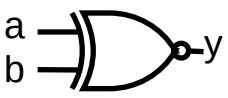
1	0	0	1
0	1	1	0
1	1	1	1
1	0	0	1

Pokrytí si snadno doplníte

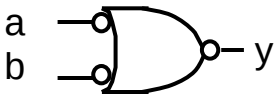
C1
.



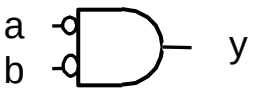
a	b	Y
0	0	1
0	1	1
1	0	1
1	1	0



a	b	Y
0	0	1
0	1	0
1	0	0
1	1	1

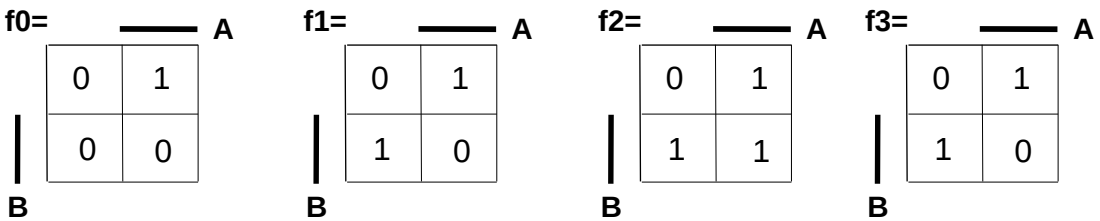


a	b	Y
0	0	0
0	1	0
1	0	0
1	1	1



a	b	Y
0	0	1
0	1	0
1	0	0
1	1	0

D3.



0	1
0	0

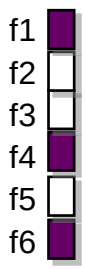
0	1
1	0

0	1
1	1

0	1
1	0

D4. *:*

f1<=(A xor C) or (A and not C);
f2<=(B or C) and (not A or B or C);
f3<=((C and not B) or (B and A));
f4<=(A or C) and (not A or not C);
f5<=(A and not B) xor (A and C);
f6<=(A and not C) or (C and not A);

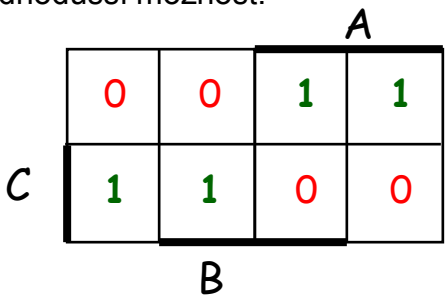


E4x- Varianta E4 – jsou uvedené funkce stejné?

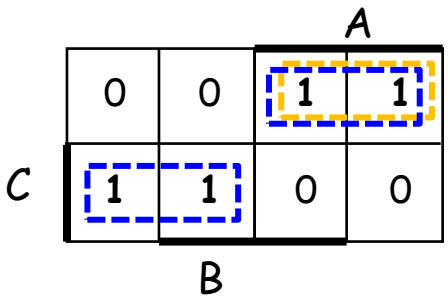
x1<=(A and not C) or (C and not A);
x2<=(A and not B) xor (A and C);
x3<=(A or C) and (not A or not C);
x4<=(A xor C) or (A and not C);

Můžete sice řešit sestavením logické tabulky z dosazováním hodnot, ale jde o zdlouhavou operaci, a zde zbytečnou, protože existuje mnohem jednodušší možnost:

X1 a x3 – se dá *řešit přímým sestavením map, protože se jedná přímo o zápisy ve formách P-o-S a S-o-P vzniklých při pokrývání K-map. Provedeme tedy jen inverzni operaci - z výrazů sestavíme mapy. Funkce x1 je S-o-P pokrytí jedniček, x3 zase P-o-S pokrytí nul.*

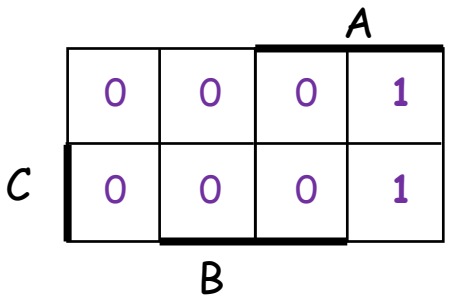


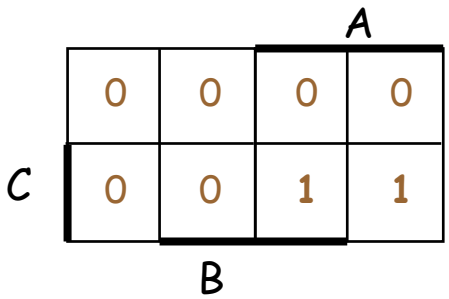
X4 *je S-o-P pokrytí jedniček, až na operaci* ***xor,*** *ale tu můžeme také graficky napsat, bude mít logické 1 jen v políčkách, kde její operandy A a C jsou různé, protože xor je non-ekvivalence. Vidíme, že člen* (A and not C) *je ve výrazu nadbytečný.*



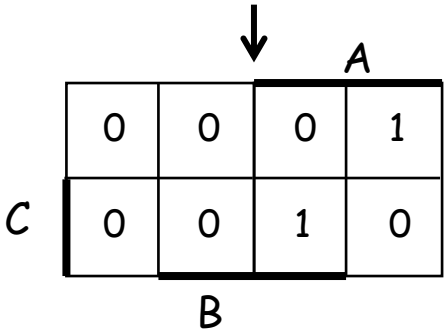
X2 obsahuje operaci také XOR, ovšem se složitějšími členy. Zde si pomůžeme sestavením K-map obou členů operace XOR.

x2<=(A and not B) xor (A and C);

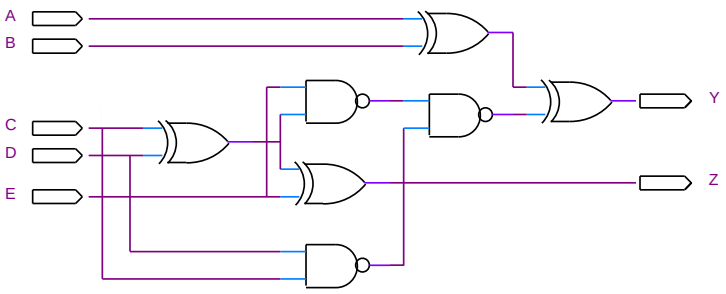


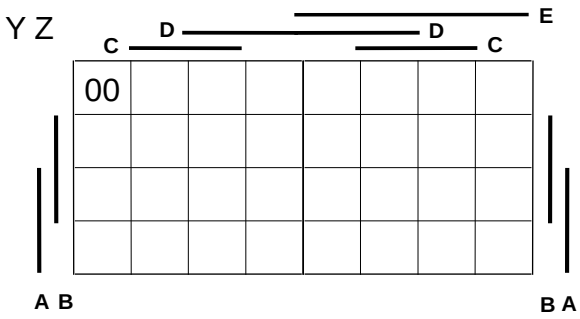


Poté provedeme jen logickou operaci xor obou K-map, jinými slovy, napíšeme 1 do jen těch políček, kde se logické hodnoty v obou mapách liší



D1

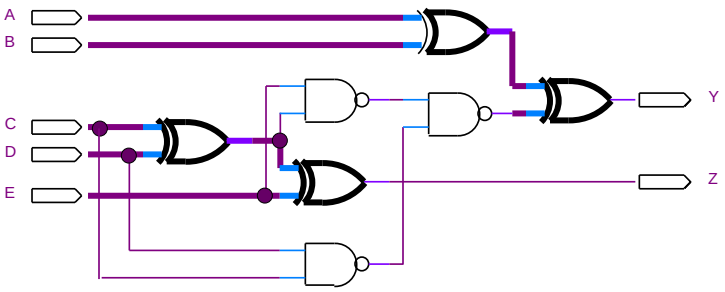


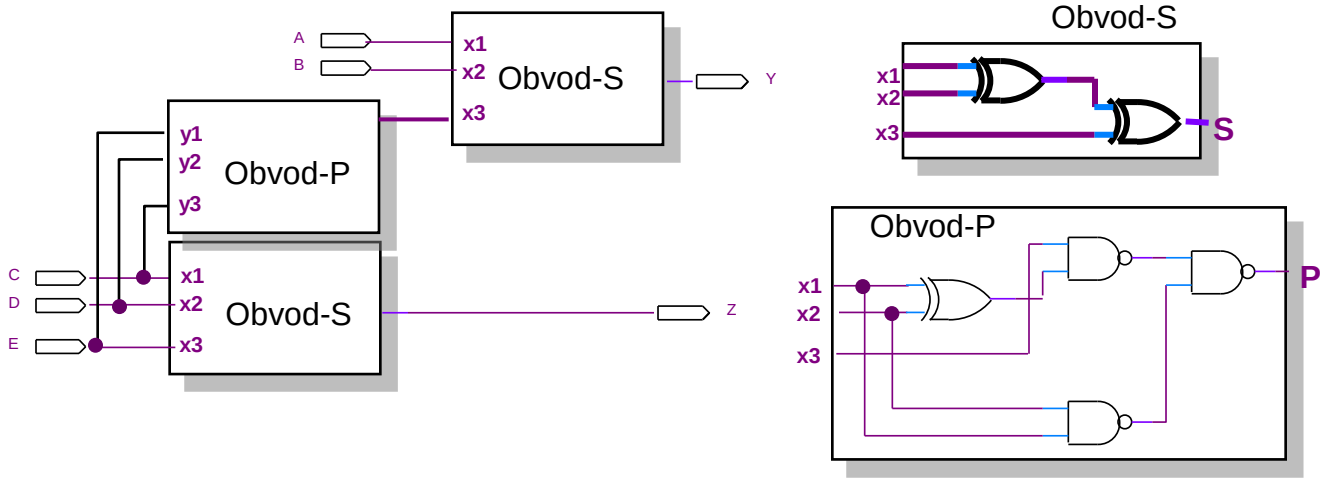


Tento "zapeklitý" příklad lze řešit na dvojí způsob - buď analogií útoku brutální silou (viz http://en.wikipedia.org/wiki/Brute-force_attack), nebo použít logickou dekompozici.

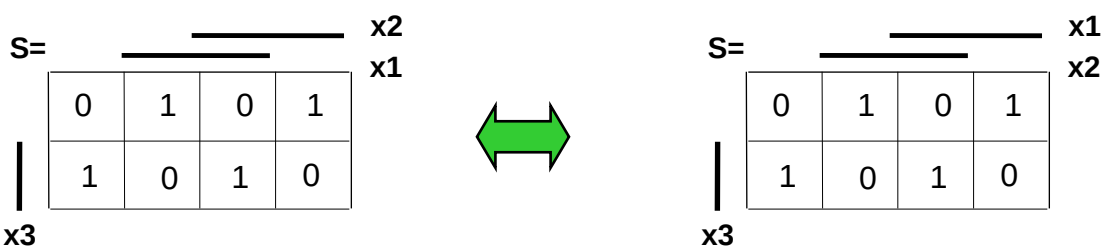
V prvém případě dosadíme všechny možné kombinace, těch je zde naštěstí jen milosrdných $2^5=32$ a získáme pravdivostní tabulku, aniž bychom porozuměli funkci obvodu:-(

V druhém případě budeme hledat analogie, symetrie a podobné části, vytvoříme jejich K-mapy, a ty skombinujeme. Najdeme dvě podobnosti:

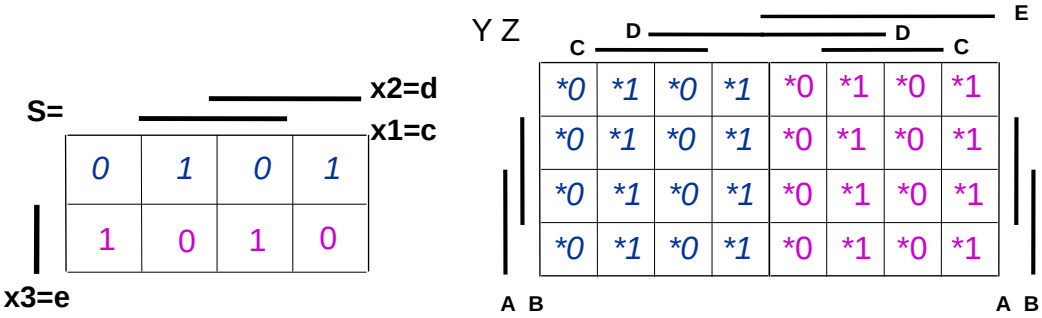




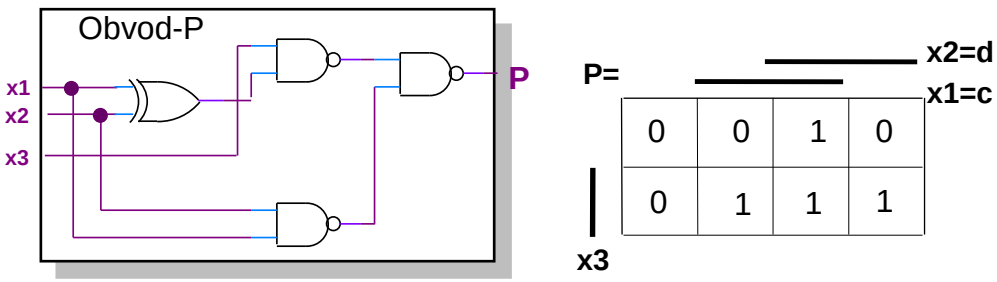
Obvod-S má rovnici **S := (x1 xor x2) xor x3**, která dává '1' při vstupních kombinacích **001, 010, 100 a 111**, tedy lichém počtu '1' na vstupech. Všimněte si, že proměnné x1 a x2 ve funkci S jsou záměnné, můžeme je prohodit podle potřeby



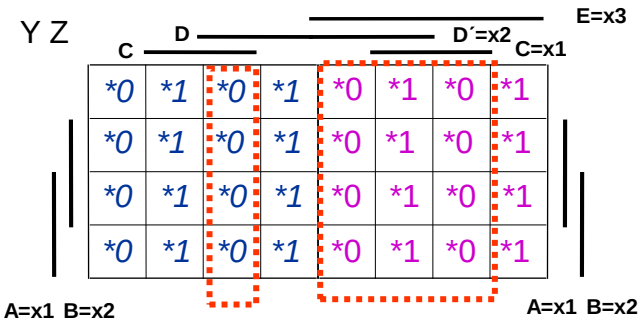
Nyní snadno vyplníme půlku pravdivostní tabulku, Z = S(c,d,e) = S(c=x1, d=x2, e=x3)



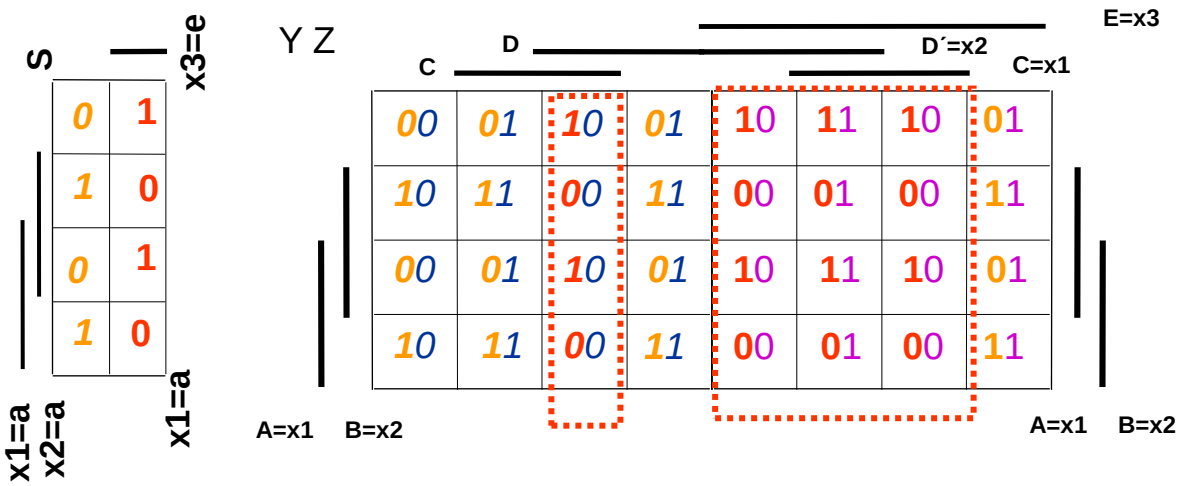
Obvod-P má rovnici R := **not**(not(x3 and (x1 xor x2)) and not(x1 and x2)). Po rozepsání zvýrazněné negace dostaneme **X:= ((x3 and (x1 xor x2)) or (x1 and x2)).** Snadno nahlédneme, jeho výstup bude v '1' pro vstupní kombinace x1 x2 x3 = 11- 101 011, tedy **110 111 101 011.** Obvod dává 1 při počtu jedniček dva a více.



V naší polovyplněné tabulce označíme sloupce, pro které obvod-P bude dávat na výstupu 1, tam bude x3 rovno 1 u vstupu horního obvodu S



Nyní použijeme opět tabulku funkci S, kde využijeme záměny x1 a x2, a pomocí ní vyplníme prvky po řádcích



D2. *Obrázku obvodu z otázky 1 chybí titulek. Správně pojmenujte jeho funkci s použitím nejvýše 7 slov (pozn. nepočítají se jednopísmenné spojky a předložky a jména signálů z obrázku).*

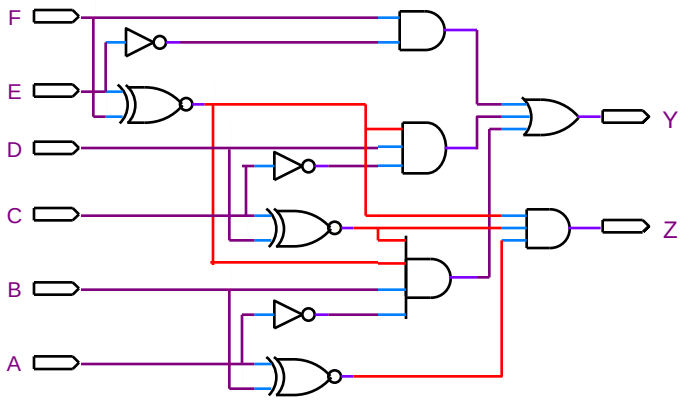
*Pokud se podíváme na řešení odpovědi na D1, pak uvidíme, že se v zapojení opakuji některé obvody - * obvod S s rovnici S := (x1 xor x2) xor x3 jen v '1' na výstupu při vstupních kombinacích 001, 010, 100 a 111, tedy lichém počtu '1' na vstupech. Je to tedy buď generátor sudé parity (doplňuje výstup na sudý počet jedniček), nebo 1bitová sčítačka s přenosem z nižšího řádu, napři Y = A plus B plus Carry.*

•obvod P funguje jako majorita ze 3, takže provádí generování přenosu do vyššího řádu - výstup(přenos) je jednička při dvou a více vstupech v jedné.

Spojíme-li obě informace dohromady, pak správný název je
dvoubitová úplná sčítačka *nebo* ***dvoubitová sčítačka s přenosem z nižšího řádu***

E1.  je operace ekvivalence ≡, tj rovnost, takže Z: číslo FDB = číslo ECA - diagonála.

Uvědomíme-li si, že (F and not E) je porovnání (F>E), čili bit F větší než bit E, pak snadno sestavíme rovnici Y = (F>E) or ((F ≡ E) and (D>C)) or ((F ≡ E) and (D ≡ C) and (B> A)), což je 3bitový komparátor "větší než", Y: číslo FDB > číslo ECA. Šachovnicovitě vyplníme K-mapu podle hierarchie bitů. Jen když jsou vyšší bity shodné, uplatní ty nižší. Vyznačeno žlutě F ≡ E, modře D ≡ C, zeleně B ≡ A



Y Z \ B D	01	10	10	10	10	10	10	10
A C	00	01	10	10	10	10	10	10
	00	00	01	00	10	10	10	10
E	00	00	10	01	10	10	10	10
	00	00	00	00	01	10	00	00
	00	00	00	00	00	01	00	00
	00	00	00	00	10	10	01	00
A	00	00	00	00	10	10	10	01

E3:

f0=

B \ A	0	1
0	0	1
1	0	0

f1=

B \ A	0	1
0	0	1
1	1	0

f2=

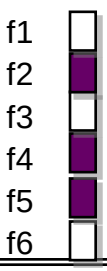
B \ A	0	1
0	0	1
1	1	1

f3=

B \ A	0	1
0	0	1
1	1	0

E4.

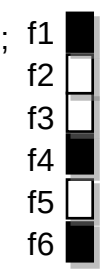
```
f1<=((C and not B) or (C and B and A));
f2<=(A xor C) or (A and not C);
f3<=(A or B) and (not A or B or C);
f4<=(not A or not C) and (C or A);
f5<=(not C and A) or (not A and C);
f6<=(A and not B) xor (A and and not B and C);
```



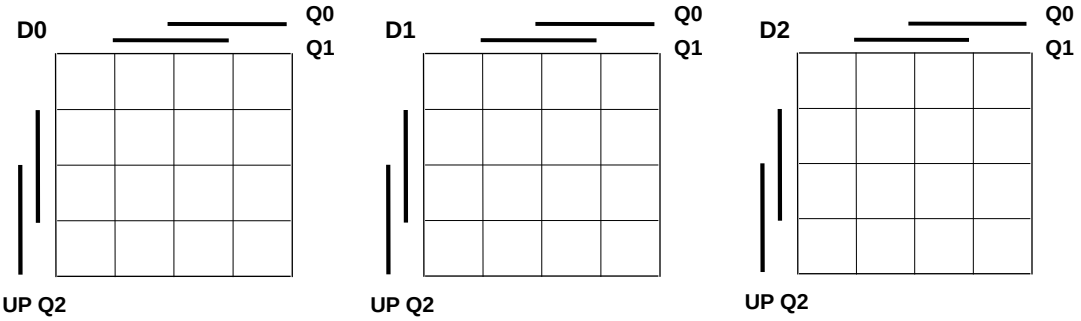
C4.

D C \ B A	0	1	2	3
0	0	0	1	0
1	1	0	1	0
2	1	1	1	0
3	0	0	1	0

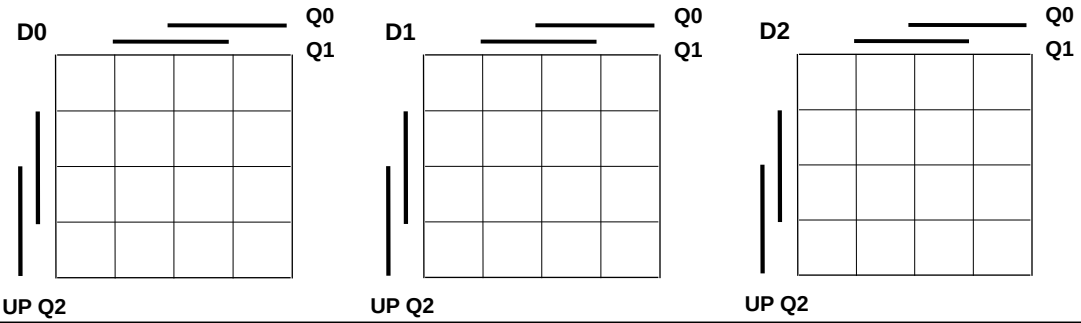
```
f1<=(((not B and not A) or (A and D)) and C) or ( B and A);
f2<=((((not A or D) and C ) or A) and B);
f3<=((C and (not A or D)) or B) and A and not B;
f4<=((not A or D) and C and not B) or ( B and A);
f5<=(A and not B) or (A and D) or ( B and A);
f6<=(B or C) and (not A or B or D) and (A or not B);
```



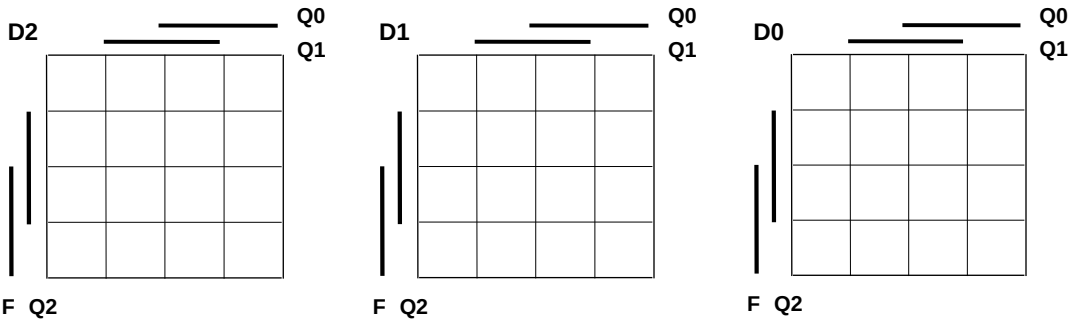
B8. Napište Karnaughovy mapy vstupů D0, D1, D2 klopných obvodů DFF pro 3bitový synchronní 6 stavový Grayův čítač (výstupy Q0, Q1, Q2 = 000 001 011 010 110 100 *000 001*... atd.) Je-li UP=1, pak se čítá nahoru (sekvence se prochází zleva doprava), při UP=0 se sekvence prochází opačně, tj. zprava doleva. Nezapomeňte na stavy X (don't care)



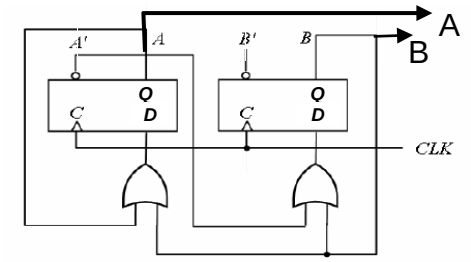
A8. Napište Karnaughovy mapy vstupů D0, D1, D2 klopných obvodů DFF pro 3bitový synchronní Johnsonův čítač (výstupy Q0, Q1, Q2 = 000 001 011 111 110 100 *000 001*... atd.) Je-li UP=1, pak se čítá nahoru (sekvence se prochází zleva doprava), při UP=0 se sekvence prochází opačně, tj. zprava doleva. Nezapomeňte na stavy X (don't care)



F7. Napište Karnaughovy mapy vstupů D0, D1, D2 klopných obvodů DFF pro 3bitový synchronní Johnsonův čítač (výstupy Q2, Q1, Q0 = 000 001 011 111 110 100 *000 001*... atd.) Je-li F=1, pak se čítá zrychleně o dvě pozice ze současného stavu, např. z 000 na 011, z 001 na 111, při F=0 se sekvence prochází po jednom členu. Nezapomeňte na stavy X (don't care)



A7. Synchronní klopné obvody D (typ DFF) na obrázku vpravo reagují na náběžné hrany signálu CLK. *Napište, jak se budou měnit výstupy A a B po příchodu náběžné hrany hodin z počátečních stavů 0 (5 bodů)*



CLK

A=	**0**						
B=	**0**						

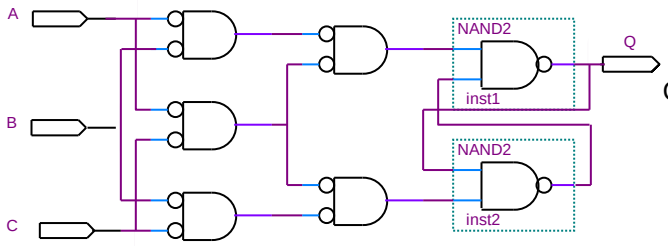
C2. Vstupy A, B, C měly v časech t_0, t_1, t_2, t_3 hodnoty uvedené v obrázku. Napište hodnoty výstupu Q. Předpokládejte, že intervaly mezi změnami vstupů jsou tak dlouhé, že lze zanedbat zpoždění hradel.

A = ..0..|..0..|..1..|..1..|

B = ..0..|..0..|..0..|..1..|

C = ..1..|..0..|..0..|..0..|

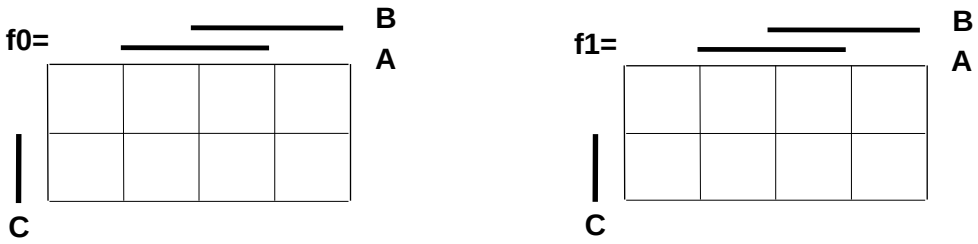
t_0..|..t_1..|.. t_2..|..t_3..|..



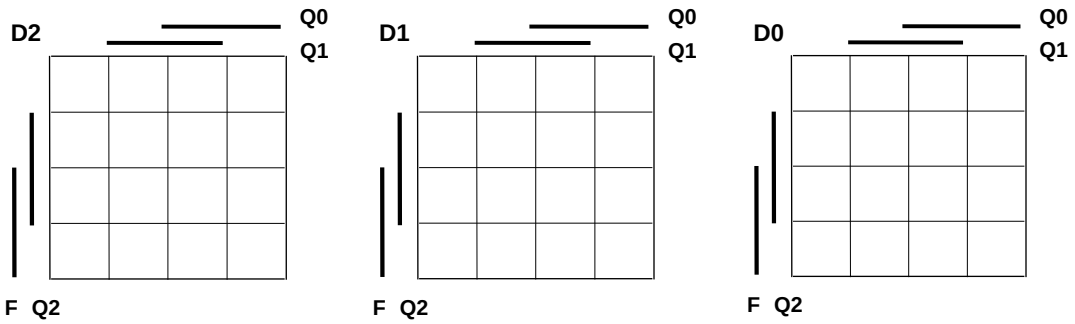
Q = ...|.....|.....|.....|

t_0..|..t_1..|..t_2..|..t_3..|..

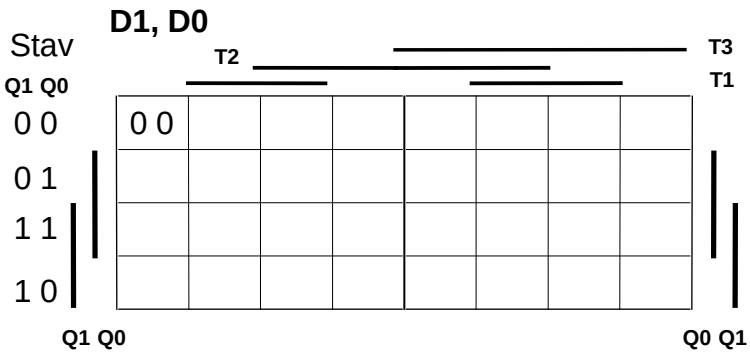
C3. Funkci Q=f(A,B,C,Q) z otázky 2 rozložte na tvar $Q=\overline{Q}.f0(A,B,C) + Q.f1(A,B,C)$ pomocí Shannonovy dekompozice. Výsledné funkce f0 a f1 napište jako Karnaughovy mapy:



C8. Napište Karnaughovy mapy vstupů D2, D1, D0 klopných obvodů DFF pro 3bitový synchronní binární čítač 0 až 5 směrem nahoru. Je-li F=1 (fast mode), pak se čítá zrychleně a sekvence binárních čísel 000 až 101 se prochází s přírůstkem 2, tedy např. z hodnoty 001 na 011 pak na 101, ale z 010 se skoči na 100, pak na 000, apod. Při F=0 se sekvence prochází s přírůstkem 1. Nezapomeňte na stavy "don't care" - ty označte jako X.



D8. Pomocí *dvou synchronních DFF klopných obvodů s hodinami 10 kHz vytvořte automat, jehož výstup Q1,Q0=00 až 11 bude hlásit nejvyšší počet současně stisknutých tlačítek T1, T2 a T3 od poslední inicializace DFF obvodů asynchronním nulováním. Vstupy D klopných obvodů DFF zapište do společné tabulky v pořadí D1 D0 (vyšší. nižší bit).*



F2. Vstupy A, B, C měly v časech t_0, t_1, t_2, t_3 hodnoty uvedené v obrázku. Napište hodnoty výstupu Q. Předpokládejte, že intervaly mezi změnami vstupů jsou tak dlouhé, že lze zanedbat zpoždění hradel.

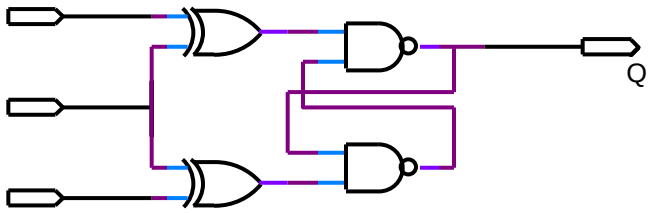
A = **..1..| 1..|..1..|..0..|**

t_0..|..t_1..|.. t_2..|..t_3..|..

B = **..0..|..1..|..1..|..1..|**

t_0..|..t_1..|.. t_2..|..t_3..|..

C = **..0..|..0..|..1..|..1..|**

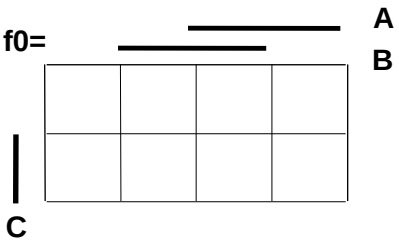


Q = ...|… ….|… ….| ……|

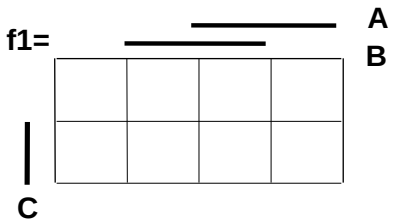
t_0..|..t_1..|.. t_2..|..t_3..|..

F3. Funkci Q=f(A,B,C,Q) z otázky 2 rozložte na tvar Q=$\overline{Q}$.f0(A,B,C) + Q.f1(A,B,C) pomocí Shannonovy dekompozice. Výsledné funkce f0 a f1 napište jako Karnaughovy mapy

f0=



f1=



A6 a) Kroužky vyznačte stabilní stavy automatu v jeho přechodové tabulce (x_1,x_2 označuje vstupy)

y_1y_2 \ x_1x_2	00	01	11	10
00	00	10	11	01
01	11	00	10	01
11	00	11	01	11
10	00	10	00	10

A6 b) Najděte všechny nestabilní cykly v přechodové tabulce automatu, (x1,x2 jsou vstupy)

y_1y_2 \ x_1x_2	00	01	11	10
00	00	10	11	01
01	11	00	10	01
11	00	11	01	11
10	00	10	00	10

E9. *Doplňte chybějící části definice*

Automat Moore (Mealy) je uspořádaná šestice **M = < X, S, Z,** ω**,** δ**,** $s_0 \in$ **S >,** kde

X je..... .....

S je.....

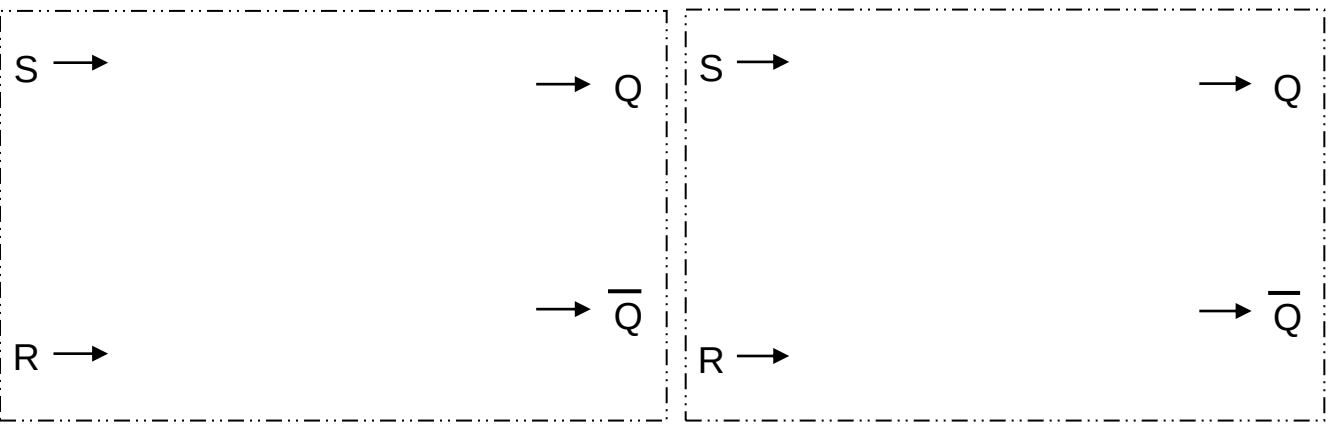
Z je.....

δ je zobrazení pro Moore..... pro Mealy.....

ω je zobrazeni pro Moore.....pro Mealy.....

s0 je .....

E6. Nakreslete klopný obvod typu R-S složený jen z hradel NOR a tentýž jen pomocí z hradel NAND



C9. *Doplňte chybějící části programu VHDL tak, aby vznikl D klopný obvod se vstupem **d** a výstupem **q**, mající asynchronní nulování při **aclrn**='0' a synchronní nulování při **sclrn**='0'.*

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity mydff is port (clock, d, aclrn, sclrn : in std_logic; q : out std_logic) end mydff;
architecture rtl of mydff is begin
  ..........
  ..........
  ..........
  ..........
  ..........
  ..........
end rtl;
```

D9. *Doplňte chybějící části programu VHDL tak, aby vznikl posuvný 100 bitový registr, tj. na jeho výstupu **q** bude vstup **d** zpožděný o 100 hodinových taktů signálu **clk**. Registr se synchronně nuluje při signálu **sclrn**='0'. Volte co nejúspornější kód. (Nápověda. ten nejkratší kód nemá instrukce cyklů).*

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity pos100 is port (clock, d, sclrn : in std_logic; q: out std_logic) end pos100;
architecture rtl of pos100 is begin
  ..........
  ..........
  ..........
  ..........
  ..........
  ..........
  ..........
end rtl;
```

E8. *Redakce technického časopisu Vás požádala, abyste rozluštili následující záhadný VHDL kód.*

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity zahadum is port (a, b, c : in std_logic; d : out std_logic); end;
architecture rtl of zahadum is signal z:std_logic_vector(0 to 2);
begin process(a, b) begin
  if a='0' then z<=(others=>'0'); elsif b'event and b='1' then z<=c & z(0 to 1); end if; d<=z(2);
end process; end rtl;
```

Pomocí hradel a klopných obvodů nakreslete čtenářům schéma logického obvodu odpovídajícího uvedenému VHDL kódu. Obvod správně pojmenujte:

A2. Mějme 16tibitovou aritmetiku se znaménkem, používající vyjádření záporných čísel ve druhém doplňku, VHDL typ signed(15 downto 0), najděte

a) číslo X, které po připočtení k číslu Y=70CA (zapsanému hexadecimálně) dá 0, tj. X+Y=0 (hledané číslo X napište opět hexadecimálně) X= .....

b) číslo X, které po připočtení k dekadickému číslo Y=100 dá 10, tedy X+Y=10 (hledané číslo X napište opět hexadecimálně) X =.....

Odpovědí na otázku A2 smí obsahovat jen hexadecimální číslice bez znmének!

B2. Mějme 8bitovou aritmetiku se znaménkem, používající vyjádření záporných čísel ve druhém doplňku, VHDL typ signed(7 downto 0) najděte

a) číslo X, které po připočtení k číslu Y=9A (zapsanému hexadecimálně) dá 0, tj. X+Y=0 (hledané číslo X napište opět hexadecimálně) X=.....

b) číslo, které po připočtení k dekadickému číslu Y=50 dá -1, tedy X+Y=-1 (hledané číslo X napište opět hexadecimálně) X =.....

Odpovědí na otázku 2 smí obsahovat jen hexadecimální číslice bez znmének!

C7. Mějme **14bitovou** aritmetiku se znaménkem, používající vyjádření záporných čísel ve druhém doplňku, VHDL typ signed(13 downto 0) Najděte číslo X, které po připočtení k číslu Y=1234 (zapsanému dekadicky) dá 0, tj. X+Y=0 Hledané číslo X napište hexadecimálně: X=3B2E.....

Odpověď na otázku 2 smí obsahovat jen hexadecimální číslice bez znamének!

Z technických důvodů - umísťování obrázků na stránce
- nejdou všechny odpovědi nutně za sebou ve stejném pořadí jako zadané příklady

B8. Napište Karnaughovy mapy vstupů D0, D1, D2 klopných obvodů DFF pro 3bitový synchronní 6 stavový Grayův čítač (výstupy Q0, Q1, Q2 = 000 001 011 010 110 100 *000 001*... atd.) Je-li UP=1, pak se čítá nahoru (sekvence se prochází zleva doprava), při UP=0 se sekvence prochází opačně, tj. zprava doleva. Nezapomeňte na stavy X (don't care).

Řešení: Čítač je konečný automat. Využije postup demonstrovaný na přednášce a popsaný ve skriptech. Napřed sestavíme přechodovou tabulku obousměrného 6tistavového čítače - první sloupec je současný stav, ostatní budoucí (následující) stavy po příchodu hodin. Tabulka zakódujeme, tj. čísla stavů nahradíme binárními kódy stavů dle zadání.

Přechodová tabulka obecného obousměrného 6tistavového čítače

Stav	UP=0	UP=1
0	5	1
1	0	2
2	1	3
3	2	4
4	3	5
5	4	0

Stav	UP=0	UP=1
000	100	001
001	000	011
011	001	010
010	011	110
110	010	100
100	110	000

Zakódovaná tabulka-zadaného čítače

Zakódovanou tabulku zadaného čítače přepíšeme do Karnaughovy mapy. Pomůžeme si tím, že si čerchovanými čárami vyznačíme pohyb při čítání. Zbylé nepoužité stavy budou X (don't care), protože správně pracující čítač v nich nikdy nebude.

Q0 Q1 Q2

UP \ Q0 Q1	00	01	11	10
0	100	011	010	110
1	000	001	x	x
0	011	010	x	x
1	001	110	100	000

UP Q2

Realizujeme-li čítač klopnými obvody typu D, pak datový vstup D klopného obvodu musí mít hodnotu následujícího výstupu. Karnaughovu mapu následujících stavů můžeme proto přímo rozepsat na mapy vstupů D0, D1, D2 - Q0 bit dáme do první mpy pro D0, Q1 bit do druhé mapy pro D1 a Q2 bit do mapy D2.

D0

UP Q2 \ Q0 Q1	00	01	11	10
0	1	0	0	1
1	0	0	x	x
0	0	0	x	x
1	0	1	1	0

D1

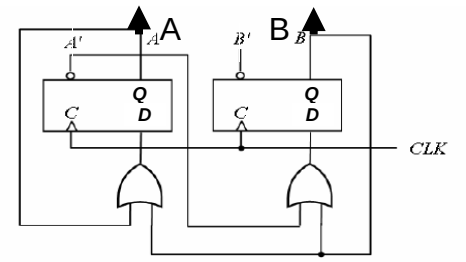
UP Q2 \ Q0 Q1	00	01	11	10
0	0	1	1	1
1	0	0	x	x
0	1	1	x	x
1	0	1	0	0

D2

UP Q2 \ Q0 Q1	00	01	11	10
0	0	1	0	0
1	0	1	x	x
0	1	0	x	x
1	0	0	0	0

Z technických důvodů - umísťování obrázků na stránce
- nejdou všechny odpovědi nutně za sebou ve stejném pořadí jako zadané příklady

A7.



CLK

A=	**0**	**0**	**1**	**1**	**1**	**1**	**1**
B=	**0**	**1**	**1**	**1**	**1**	**1**	**1**

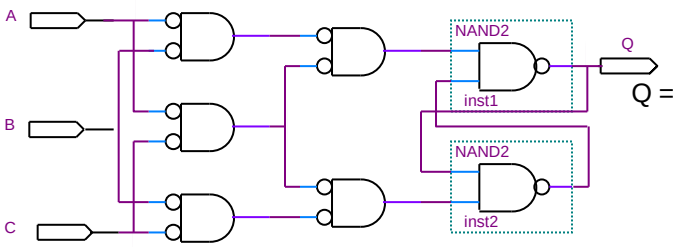
C2.

A = ..0..|..0..|..1..|..1..|

B = ..0..|..0..|..0..|..1..|

C = ..1..|..0..|..0..|..0..|..0..|

t_0..|..t_1..|.. t_2..|..t_3..|..



Q = ...1...|…1…|…0…|…0….|

t_0..|..t_1..|..t_2..|..t_3..|..

C3.

f0=

C \ B A	00	01	11	10
0	1	0	0	1
1	1	0	0	0

f1=

C \ B A	00	01	11	10
0	1	0	1	1
1	1	1	1	1

C8..

D2

F Q2 \ Q0 Q1	00	01	11	10
0	0	0	1	0
1	1	X	X	0
0	0	X	X	0
0	0	1	1	0

D1

F Q2 \ Q0 Q1	00	01	11	10
0	0	1	0	1
0	0	X	X	0
0	0	X	X	0
1	1	0	0	1

D0

F Q2 \ Q0 Q1	00	01	11	10
1	1	1	0	0
1	1	X	X	0
0	0	X	X	1
0	0	0	1	1

Z technických důvodů - umísťování obrázků na stránce
- nejdou všechny odpovědi nutně za sebou ve stejném pořadí jako zadané příklady

F2

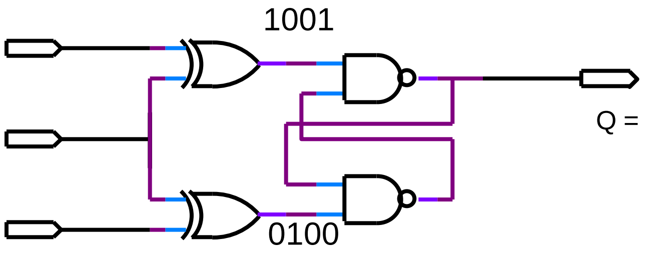
A = **..1..| 1..|..1..|..0..|**

t_0..|..t_1..|.. t_2..|..t_3..|..

B = **..0..|..1..|..1..|..1..|**

t_0..|..t_1..|.. t_2..|..t_3..|..

C = **..0..|..0..|..1..|..1..|**



Q = ...0 ...|... 1....|...1 ....| ...0...|

t_0..|..t_1..|.. t_2..|..t_3..|..

Podobné úlohy, obsahující klopné obvody, se nejlépe řeší tak, že si uděláme posloupnost signálů na vstupu klopného obvodu a poté si vzpomeneme, jak klopný obvod funguje. non-RS nakreslený ve schématu je ovládaný logickými nulami. Vstupy S=1 R=0 nastaví Q do 0, Vstupy S=0 R=1 nastaví Q do 1, hodnoty vstupů S=0 R=0 přivedou výstupy do zakázaného stavu, kdy Q=1 i nonQ=1 a neví se následující stav, ale naštěstí opět následuje S=1 R=0, takže Q bude 0.

F3

f0= (A, B, C)

1	0	1	0
1	0	1	0

f1= (A, B, C)

1	1	1	0
1	0	1	1

A2. a) X= **8F36**...

b) X =**FFA6**...

B2. a) X=66...

b) X=0CD...

C7 X=3B2E.......- výsledkem musí být 14 bitové číslo!!!!....................................

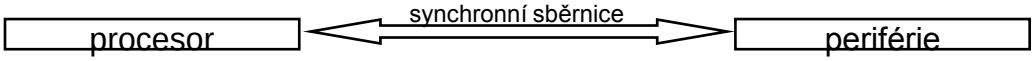
C10. Mějme procesor s 7 jednobitovými registry označenými Q0 až Q5 a Y, kde registr Y slouží současně jako výstup procesoru. Po spuštění procesoru se všechny registry vynulují a poté se spustí nekonečná programová smyčka. Symbol ← znamená přiřazení hodnoty:

Loop: Y ← Q5; Q5 ← Q4; Q4 ← Q3; Q3 ← Q2; Q2 ← Q1; Q1 ← Q0; Q0 ← Y xor 1;
Jump **Loop;**

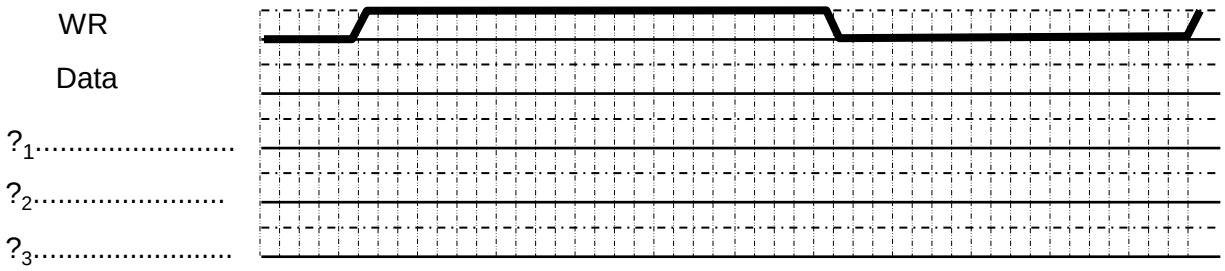
Napište hodnoty, které bude mít registr Y na začátcích cyklu Loop hned po vykonání operace Y ← Q5:

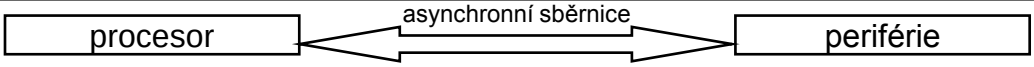
cyklus	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.
Y=	***0***																

12-13. přednáška



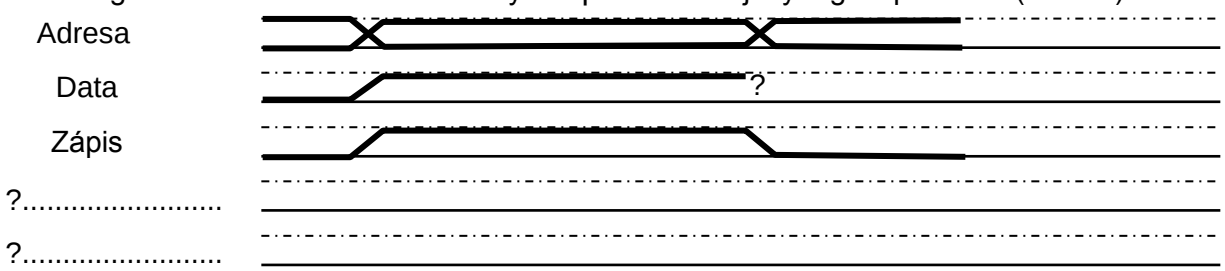
B10. Doplňte chybějící signály a průběhy na synchronní sběrnici, kde signál WR='1' určuje zápis dat do periférie a WR='0' znamená čtení dat z periférie; Data je obousměrný signál, napřed se Data zapisují do periférie, poté se z ní čtou. Čáry v obrázku jsou pomocné kreslicí linky.





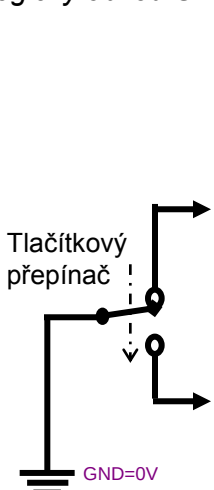
12-13. přednáška

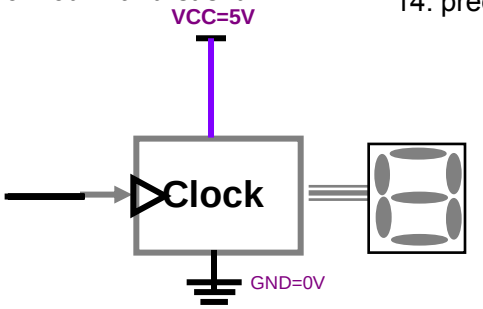
A10. Doplňte chybějící signály a průběhy na asynchronní sběrnici, kde signál Zápis='1' určuje zápis dat do periférie, Zápis='0' znamená čtení dat z periférie; Data je obousměrný signál a signál "Adresa" v obrázku dole vybírá pokaždé nějaký registr periférie. (5 bodů)



D5. *V zapojení dokreslete součástky potřebné k připojení mechanického tlačítkového přepínače na hodinový vstup dekadického čítače počítajícího počet stisků přepínače. Na vstupu čítače je logický obvod CMOS mající Vcc=5 V a čítač se má klopit na náběžnou hranu stisku.*

14. přednáška

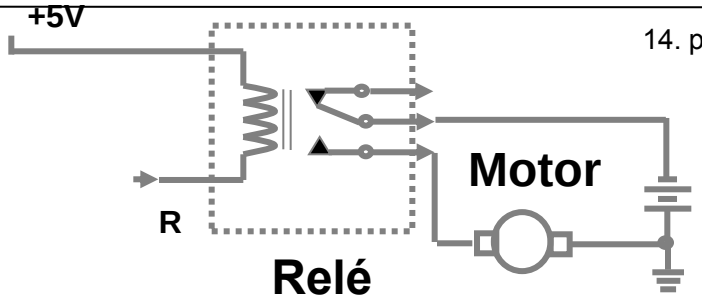


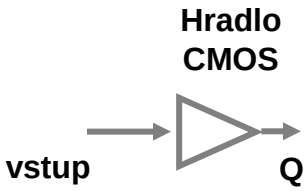


Dekadický cítač

C5. V zapojení dokreslete součástky potřebné k propojení výstupu Q z hradla CMOS s Vcc=5 V ke vstupu R relé spínajícího motor.

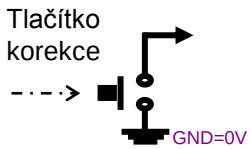
14. přednáška

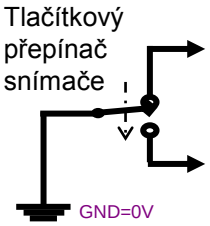


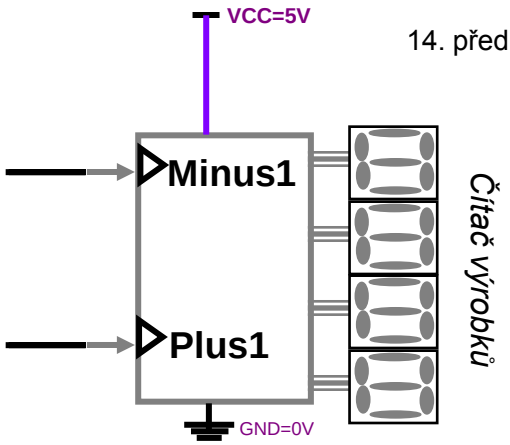


F7. Na výrobní lince je snímač výrobků provedený jako tlačítkový přepínač. Při manuální technické kontrole, může být výrobek vyřazen, což se hlásí pouhým tlačítkem. Tlačítka aktivuji hodinové vstupy registru výrobků, které reagují na náběžné hrany, snímač vstup Plus1, korekce vstup Minus1. *Doplňte mezi tlačítka a čítač potřebné součástky.*

14. přednáška

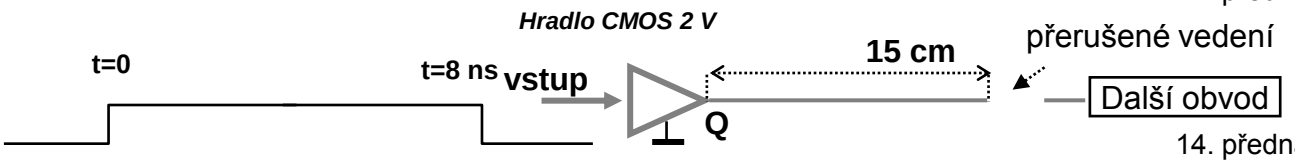




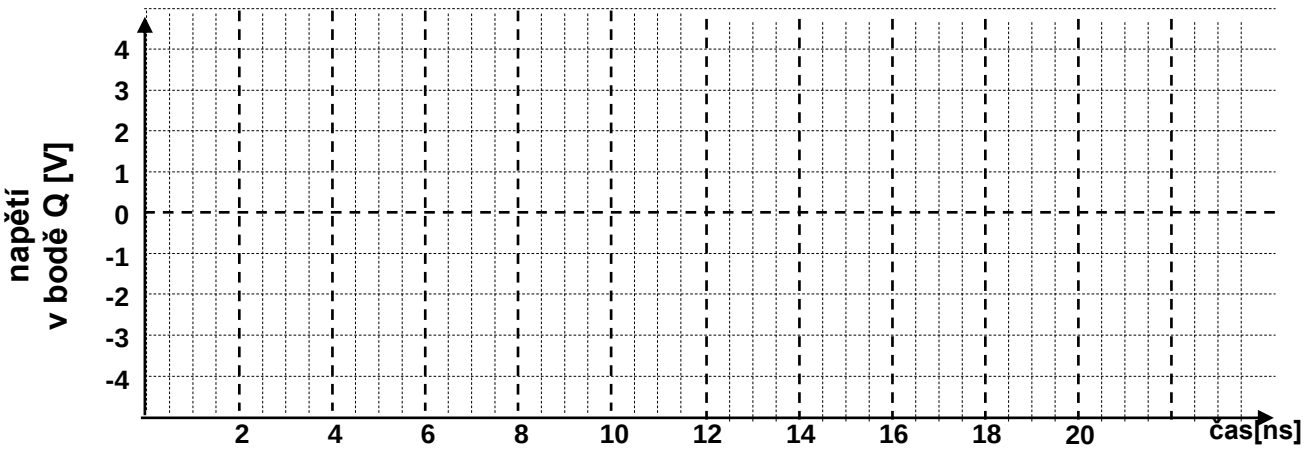


F9. *Za rychlým hradlem (budičem) s Vcc=2 V došlo nešťastnou náhodou k přerušení vedení, a to ve vzdálenosti 15 cm za hradlem. Odhadněte průběh napětí na výstupu Q hradla, pokud v čase t=0 přejde vstup hradla z 0 V do 2 V na dobu 8 ns a poté se vrátí zpět do 0. Zpoždění hradla je 1 ns.*

14. přednáška



14. přednáška



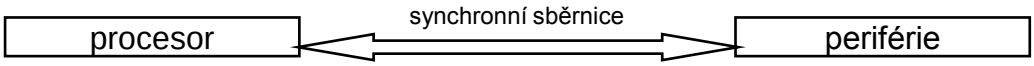
F10. *Zaškrtnutím vyznačte, která z uvedených technologii pro programovatelné obvody FPGA má*

a) nejvyšší odolnost proti radiaci	☐ Antifuse	☐ EEPROM	☐ Flash	☐ SRAM
b) nejvyšší počet možných přeprogramování	☐ Antifuse	☐ EEPROM	☐ Flash	☐ SRAM
c) nejmenší odběr ze zdroje	☐ Antifuse	☐ EEPROM	☐ Flash	☐ SRAM

Z technických důvodů - umísťování obrázků na stránce
- nejdou všechny odpovědi nutně za sebou ve stejném pořadí jako zadané příklady

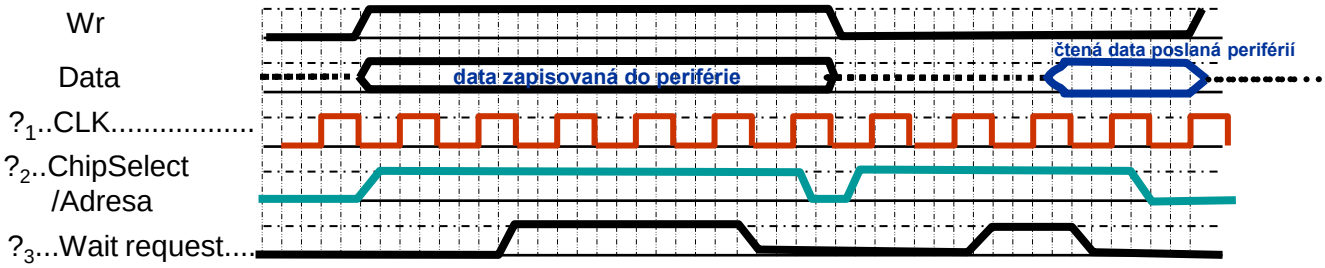
C10.

cyklus	0.	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.
Y=	***0***	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0



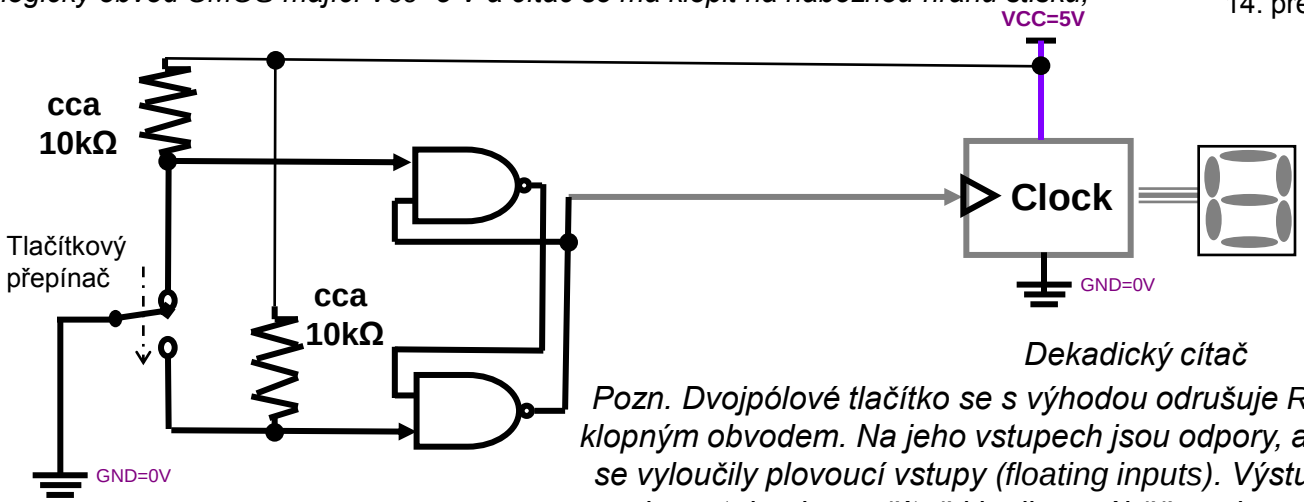
B10. Doplňte chybějící signály a průběhy na synchronní sběrnici, kde signál WR='1' určuje zápis dat do periférie a WR='0' znamená čtení dat z periférie; Data je obousměrný signál, napřed se Data zapisují do periférie, poté se z ní čtou. Čáry v obrázku jsou pomocné kreslicí linky.

Řešení: Synchronní sběrnice, jak název napovídá, musí mít hodiny. Ty budou prvním signálem. Pro zápis do periférie (zařízení slave) potřebujeme adresu nebo dekódovanou adresu (např. chip select). Jelikož přenos probíhá v dohodnutém počtu cyklů, dole 2, může periférie požádat o přidání cyklů pomocí signálu WaitRequest. Všechny signály se vzorkují aktivní hranou hodin, dole náběžnou, a musí být platné před jejím příchodem. Místo WaitRequest se samozřejmě mohou použít i jiné signály ukazované na přednášce 12, např. adresa bytu..



D5. *V zapojení dokreslete součástky potřebné k připojení mechanického tlačítkového přepínače na hodinový vstup dekadického čítače počítajícího počet stisků přepínače. Na vstupu čítače je logický obvod CMOS mající Vcc=5 V a čítač se má klopit na náběžnou hranu stisku,*

14. přednáška



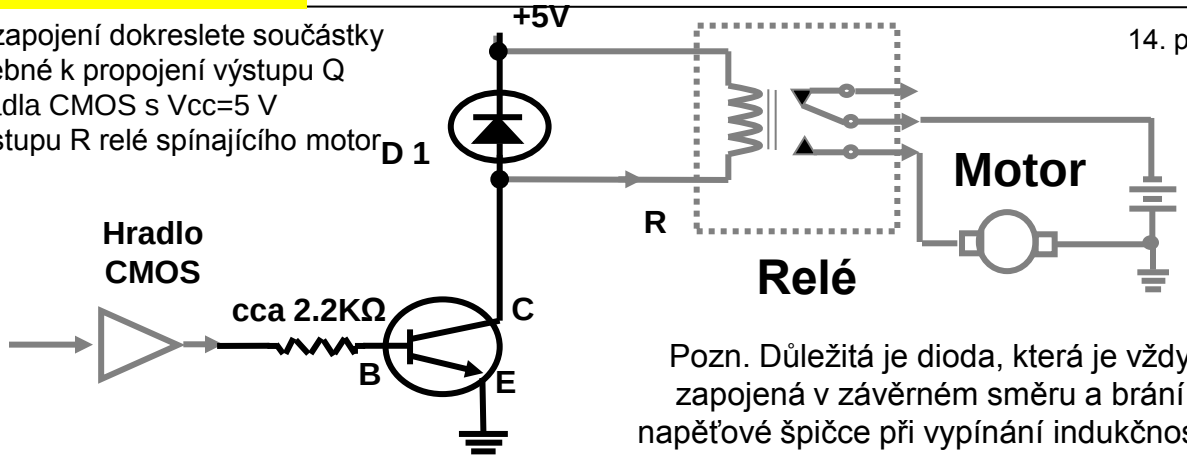
Dekadický cítač

Pozn. Dvojpólové tlačítko se s výhodou odrušuje RS klopným obvodem. Na jeho vstupech jsou odpory, aby se vyloučily plovoucí vstupy (floating inputs). Výstup vedeme tak, aby se čítač klopil na náběžnou hranu.

F10. *Zaškrtnutím vyznačte, která z uvedených technologii pro programovatelné obvody FPGA má*

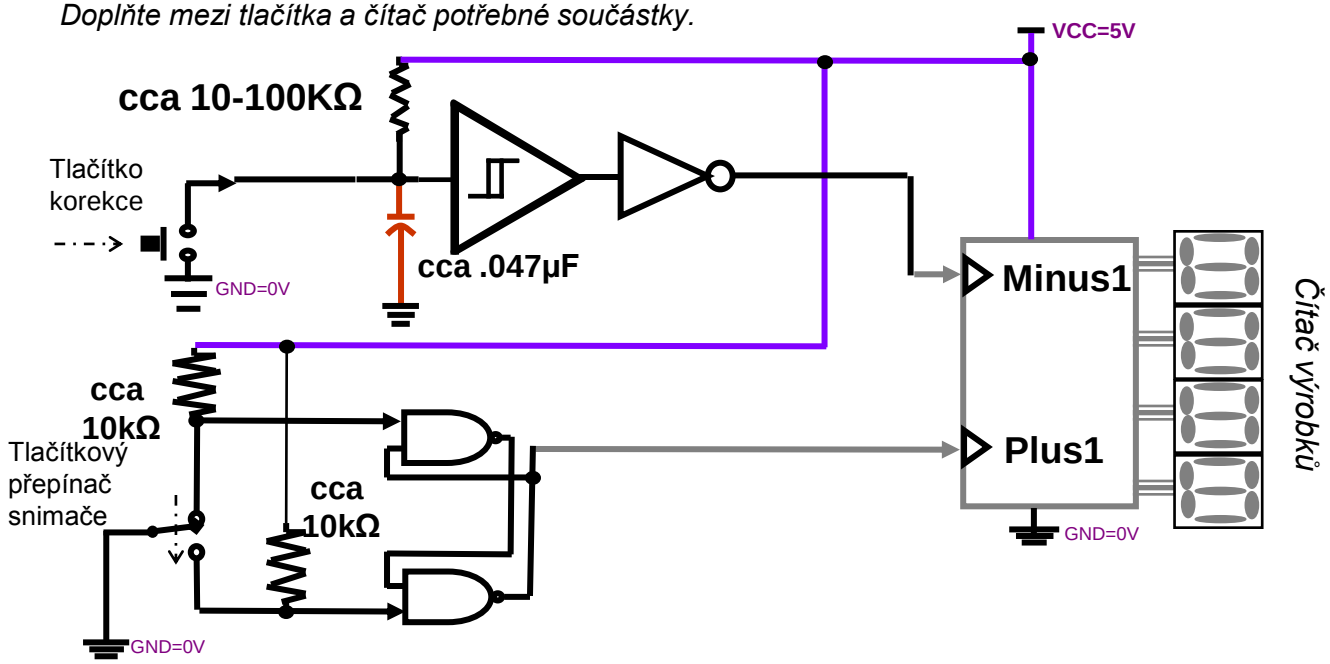
a) nejvyšší odolnost proti radiaci	■ Antifuse	☐ EEPROM	☐ Flash	☐ SRAM
b) nejvyšší počet možných přeprogramování	☐ Antifuse	☐ EEPROM	☐ Flash	■ SRAM
c) nejmenší odběr ze zdroje	☐ Antifuse	☐ EEPROM	■ Flash	☐ SRAM

C5. V zapojení dokreslete součástky potřebné k propojení výstupu Q z hradla CMOS s Vcc=5 V ke vstupu R relé spínajícího motor.



Pozn. Důležitá je dioda, která je vždy zapojená v závěrném směru a brání napěťové špičce při vypínání indukčnosti.

F7. Na výrobní lince je snímač výrobků provedený jako tlačítkový přepínač. Při manuální technické kontrole, může být výrobek vyřazen, což se hlásí pouhým tlačítkem. Tlačítka aktivuji hodinové vstupy registru výrobků, které reagují na náběžné hrany, snímač vstup Plus1, korekce vstup Minus1. *Doplňte mezi tlačítka a čítač potřebné součástky.*



Pozn. Stejně jako v D5 odrušíme dvojpólové tlačítko s výhodou RS klopným obvodem s odpory na vstupech, aby se vyloučily plovoucí vstupy (floating inputs). Výstup vedeme z RS obvodu tak, aby se čítač klopil na náběžnou hranu.

Jednopólové tlačítko musíme odrušit kondenzátorem (integračním členem), za kterým umístíme tvarovací Schmittův klopný obvod, snímek 117-120 ze 14 přednášky, abychom vyloučili pomalé náběžné hrany. Schmittův obvod obsahuje dva invertory v sérii, takže jako celek neinvertuje. Přidáme za něj tedy ještě invertor pro dosažení klopení na náběžnou hranu.

F9. *Za rychlým hradlem (budičem) s Vcc=2 V došlo nešťastnou náhodou k přerušení vedení, a to ve vzdálenosti 15 cm za hradlem. Odhadněte průběh napětí na výstupu Q hradla, pokud v čase t=0 přejde vstup hradla z 0 V do 2 V na dobu 8 ns a poté se vrátí zpět do 0. Zpoždění hradla je 1 ns.*

*Odpověď na otázku naleznete v přednáškách v části o vedení, kde je vše i s průběhy. Zde si nutno hlavně pamatovat, že přerušený vodič odráží **napěťový skok** zpět ve stejné polaritě, a ten se bude přičítat k původní hodnotě napětí, zatímco hradlo se chová jako zdroj, tedy svým způsobem jako nulový odpor, odraz od něho bude mít opačnou polaritu a bude se odečítat. Odrazy mají pochopitelně útlum, jsou vždy o něco menší než původní skok. Dále potřebujeme znát už jen rychlost světla, která činí 30 cm za nanosekundu.*

I1. Mějme VHDL deklaraci **signal X, Y, S:signed(9 downto 0);** Pokud je dekadická hodnota X rovná 500 a dekadická hodnota Y = 400, jakou hodnotu bude mít součet S=X+Y. Výslednou hodnotu S napište jako dekadické číslo S=

I2. *Funkci Q = ((*A xor B) and (D xor C)) or ((not A and B) xor (not D and C)) *rozložte na tvar* Q=$\overline{C}$.$\overline{D}$.f0(A,B) + C.$\overline{D}$.f1(A,B) + $\overline{C}$.D.f2(A,B) + C.D.f3(A,B) *pomocí Shannonovy expanze. Výsledné funkce f0, f1, f2 a f3 napište jako Karnaughovy mapy*:

f0=

B \ A	0	1
0		
1		

f1=

B \ A	0	1
0		
1		

f2=

B \ A	0	1
0		
1		

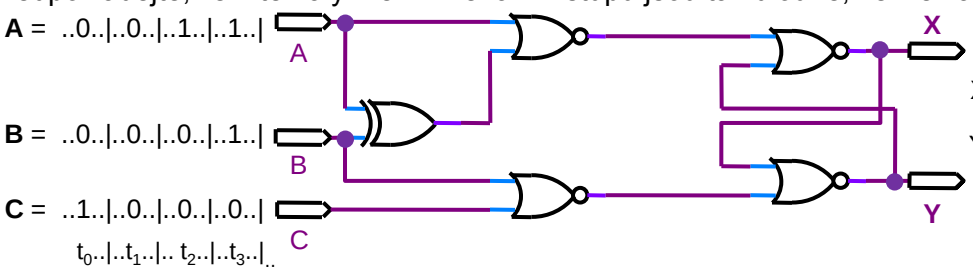
f3=

B \ A	0	1
0		
1		

Pozn. Čára symbolizuje negaci proměně jen v logických výrazech. V Karn. mapách znamená, že příslušná proměnná je v logické1.

I3. *K funkci z otázky 2 nakreslete její BDD -binary decision diagram- pořadí proměnných volte D C B A*

I4. Vstupy A, B, C měly v časech t_0, t_1, t_2, t_3 hodnoty uvedené v obrázku. Napište hodnoty výstupů X a Y. Předpokládejte, že intervaly mezi změnami vstupů jsou tak dlouhé, že lze zanedbat zpoždění hradel.

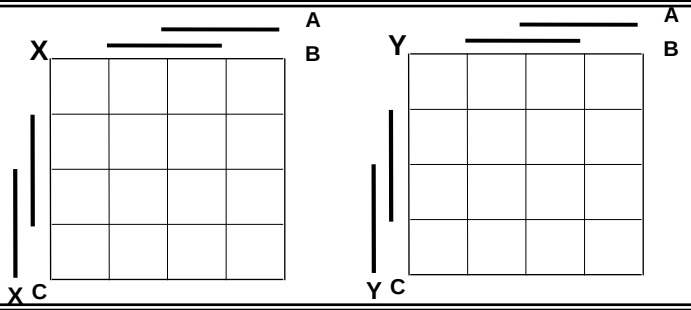


X = ...|.....|.....|.....|

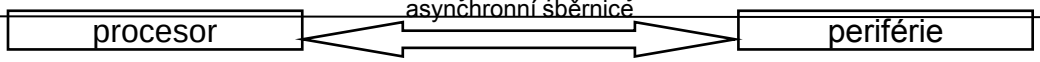
Y = ...|.....|.....|.....|

t_0..|...t_1..|.. t_2..|..t_3..|..

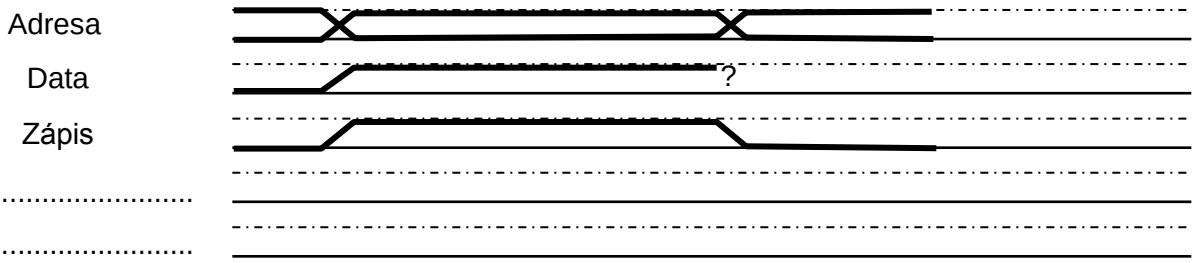
I5. Pro výstupy X a Y z příkladu 4 napište jejich Karnaughovy mapy, tj. mapy funkcí X = f(A, B, C, X)
Y = f(A, B, C, Y)



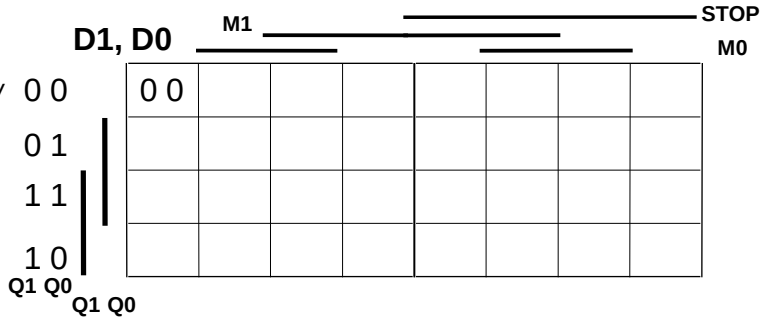
asynchronní sběrnice



I6. Doplňte chybějící signály a průběhy na asynchronní sběrnici, kde signál Zápis='1' určuje zápis dat do periférie, Zápis='0' znamená čtení dat z periférie; Data je obousměrný signál a signál "Adresa" v obrázku dole vybírá pokaždé nějaký registr periférie.



I7. Ze *dvou synchronních DFF klopných obvodů s hodinami 10 kHz a se vstupy D1 a D0 vytvořte automat, jehož stavové bity Q1 a Q0 spínají 2 motory tak, že vždy běží nejvýše jeden motor, Běží-li již jeden motor, druhý nelze zapnout. Tlačítko M1=1 zapne motor 1 (Q1,Q0=10) a tlačítko M0=1 motor 0 (Q1,Q0=01). Tlačítko STOP=1 zastaví kterýkoliv běžící motor bez ohledu na M1 a M0. (Q1,Q0=00) Pozn. Nezapomeňte na stavy X (don't care)*



8

I8. *Redakce technického časopisu Vás požádala, abyste rozluštili následující záhadný VHDL kód.*

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity zahadum is port (a, b, c, d : in std_logic; e : out  std_logic_vector(1 downto 0)); end;
architecture rtl of zahadum is signal z:std_logic_vector(1 downto 0);
begin process(d, c) begin
  if d='0' then z<=(others=>'1'); elsif c'event and c='1' then z<=b & a; end if; e<=z;
end process; end rtl;
```

Pomocí hradel a klopných obvodů nakreslete čtenářům schéma logického obvodu odpovídajícího uvedenému VHDL kódu. Obvod správně pojmenujte v titulku pod obrázkem:

I1. Mějme VHDL deklaraci **signal X, Y, S:signed(9 downto 0);** Pokud je dekadická hodnota X rovná 500 a dekadická hodnota Y = 400, jakou hodnotu bude mít součet S=X+Y. Výslednou hodnotu S napište jako dekadické číslo S=

Úlohu lze řešit na trojí způsob:

A. převedeme čísla 400 a 500 na 10bitová binární čísla, sečteme je a výsledek převedeme zpět.

```
          400 = 01 1001 0000
          500 = 01 1111 0100
      400+500 = 11 1000 0100   →  výsledek má 1 v nejvyšším bitu,
                                  jde tedy o záporné číslo, jehož absolutní
                                  hodnotu zjistíme pomocí druhého doplňku
1. doplněk:not = 00 0111 1011
     2. not+1 = 00 0111 1011
              +00 0000 0001
          124 = 00 0111 1100   →  jak tohle číslo převedeme, buď hrubou silou,
                                  nebo úvahou, kdyby mělo jedničky v posledních
                                  bitech 00 0111 1111, šlo by o 127, čili
                                  00 0111 1100 je 127-3
```

výsledek sčítání 400+500 v 10bitové aritmetice se znaménkem bude tedy -124

B. Binární sčítačka sčítá stejně čísla se znaménkem i bez znaménka, pouze jinak se pak interpretuje její výsledek. Pokud sečteme 400+500 pak v aritmetice bez znaménka dostaneme výsledek 900, ten se však bude v typu signed interpretovat jako 10bitové číslo se znaménkem. Převedeme tedy 900 na binární číslo a pro něj najdeme pomocí druhého doplňku absolutní hodnotu záporného čísla. 900 převedeme třeba dělením 2

900 / 2 - zbytek po dělení	0
450 / 2 - zbytek po dělení	0
225 / 2 - zbytek po dělení	1
112 / 2 - zbytek po dělení	0
56 / 2 - zbytek po dělení	0
28 / 2 - zbytek po dělení	0
14 / 2 - zbytek po dělení	0
7 / 2 - zbytek po dělení	1
3 / 2 - zbytek po dělení	1
1 / 2 - zbytek po dělení	1

LSB ↑ … ↑ MSB

Operací jsem dostali přímo 400+500 = **11 1000 0100 ,**
který již převedeme přes doplněk stejným postupem jako v A, čímž dostaneme -124

C. Příklad lze řešit i trojčlenkou - 10 bitová aritmetika se znaménkem má rozsah zobrazitelných čísel od -512 až 511, tj. od 10 0000 0000 až 01 1111 1111 . Víme, že 511+1, tedy 10 0000 0000, se v ní zobrazí jako -512 a 11 1111 1111 bude zobrazováno jako -1. V aritmetice bez znaménka by 11 1111 1111 mělo hodnotu1023.
Nyní máme vše potřebné pro trojčlenu s nepřímou úměrou

unsigned 512 ……je …….. -512 signed
unsigned 1023 je -1 signed

Kolik bude unsigned 900 jako signed ? 1023-900 = 123 -1 - 123 = **-124**

I2. *Funkci Q = ((*A xor B) and (D xor C)) or ((not A and B) xor (not D and C))
rozložte na tvar Q=$\overline{C}.\overline{D}$.f0(A,B) + C.$\overline{D}$.f1(A,B) + $\overline{C}$.D.f2(A,B) + C.D.f3(A,B) *pomocí Shannonovy expanze. Výsledné funkce f0, f1, f2 a f3 napište jako Karnaughovy mapy*:

f0=

B \ A	0	1
0	0	0
1	1	0

f1=

B \ A	0	1
0	1	1
1	1	1

f2=

B \ A	0	1
0	0	1
1	1	0

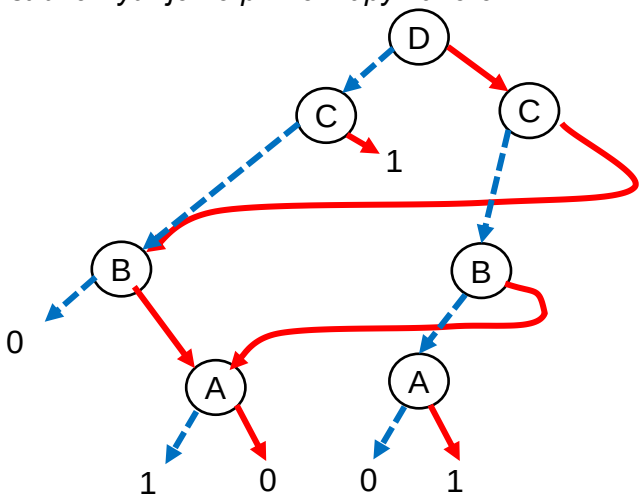
f3=

B \ A	0	1
0	0	0
1	1	0

Pozn. Čára symbolizuje negaci proměně jen v logických výrazech. V Karn. mapách znamená, že příslušná proměnná je v logické1.

I3. *K funkci z otázky 2 nakreslete její BDD - binary decision diagram - pořadí proměnných volte D C B A.*

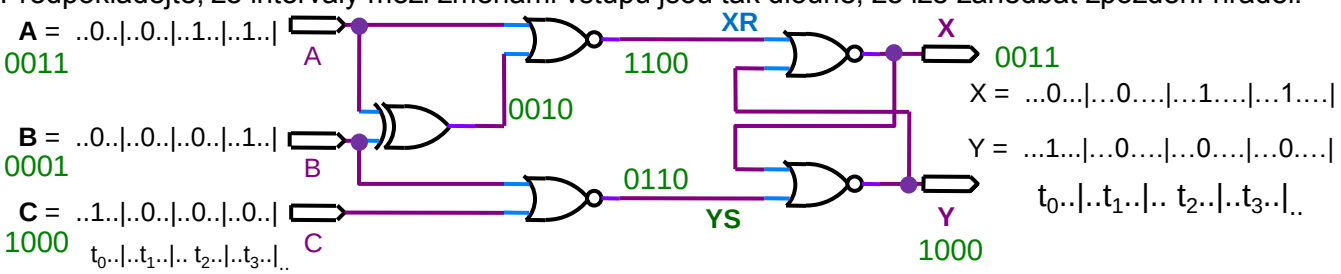
Ke konstrukci využijeme přímo mapy nahoře.



Pozn.
1/ Popis BDD najdete v přednáškách.

2/ Diagram by šel ještě dále minimalizovat tak, že by se i uzly 0 a 1 sloučily, aby se vyskytovaly pouze jedenkrát. Pro přehlednost bývá lepší při kresbě na papír nechat je raději oddělené

I4. Vstupy A, B, C měly v časech t_0, t_1, t_2, t_3 hodnoty uvedené v obrázku. Napište hodnoty výstupů X a Y. Předpokládejte, že intervaly mezi změnami vstupů jsou tak dlouhé, že lze zanedbat zpoždění hradel.



I5. Pro výstupy X a Y z příkladu 4 napište jejich Karnaughovy mapy, tj. mapy funkcí X = f(A, B, C, X)
Y = f(A, B, C, Y)

Ke konstrukci map můžeme použít buď logické výrazy pro výstupy X a Y, které si najdeme, nebo si můžeme napočítat mapy vstupů klopného obvodu, na schématu označené jako XR (reset) a YS (set). Poté vyplníme mapy X a Y podle tabulky RS klopného obvodu z hradel NOR. Jsou-li oba vstupy XS a YS ve stavech 0, tj. obvod si pamatuje, pak je všude v mapě 1 tam, kde X=1, a 0, kde X=0 - tj. žádná změna v následujícím stavu. Totéž pro Y.

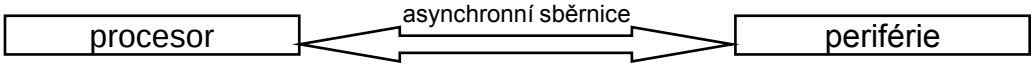
X

X C \ A B	00	01	11	10
0	0	0	0	1
1	0	0	0	0
0	0	1	1	1
1	0	1	1	1

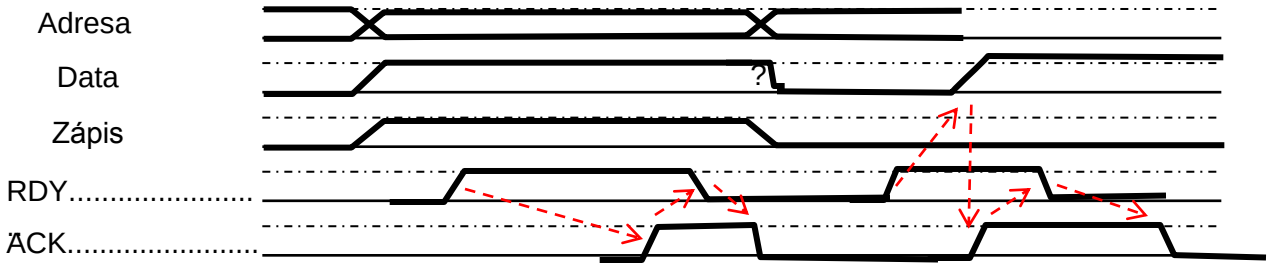
Y

Y C \ A B	00	01	11	10
0	0	0	0	0
1	1	0	0	0
0	1	1	1	1
1	0	1	1	0

YS	XR	X	Y
0	0	$X(t)$	$Y(t)$
0	1	0	1
1	0	1	0
1	1	0	0



I6. Doplňte chybějící signály a průběhy na asynchronní sběrnici, kde signál Zápis='1' určuje zápis dat do periférie, Zápis='0' znamená čtení dat z periférie; Data je obousměrný signál a signál "Adresa" v obrázku dole vybírá pokaždé nějaký registr periférie.



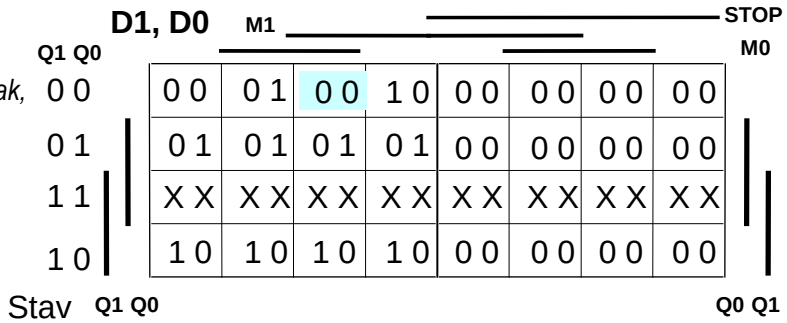
RDY zařízení1-master hlásí připravenost dat k zápis či připravenost přečíst si data

ACK zařízení2-slave potvrzuje přečtení dat či poslaní dat na sběrnici

Červené šipky udávají souslednost mezi signály

I7. Ze *dvou synchronních DFF klopných obvodů s hodinami 10 kHz a se vstupy D1 a D0 vytvořte automat, jehož výstupy Q1 a Q0 spínají 2 motory tak, že vždy běží nejvýše jeden motor, Běží-li již jeden motor, druhý nelze zapnout. Tlačítko M1=1 zapne motor 1 (Q1=1) a tlačítko M0=1 zapne motor 0 (Q0=1). Tlačítko STOP=1 zastaví kterýkoliv běžící motor bez ohledu na stav M1 a M0.*

Pozn. Nezapomeňte na stavy X (don't care)



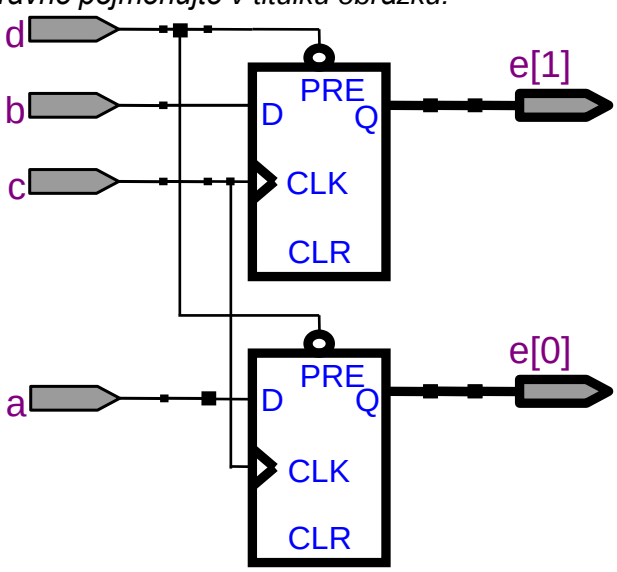
Ve zdůrazněném políčku může být i hodnota 01 nebo 10 - reakce na současný stisk. více tlačítek je na nás.

I8. *Redakce technického časopisu Vás požádala, abyste rozluštili následující záhadný VHDL kód.*

```
library IEEE; use IEEE.STD_LOGIC_1164.all;
entity zahadum is port (a, b, c, d : in std_logic; e : out  std_logic_vector(1 downto 0)); end;
architecture rtl of zahadum is signal z:std_logic_vector(1 downto 0);
begin process(d, c) begin
  if d='0' then z<=(others=>'1'); elsif c'event and c='1' then z<=b & a; end if; e<=z;
end process; end rtl;
```

Pomocí hradel a klopných obvodů nakreslete čtenářům schéma logického obvodu odpovídajícího uvedenému VHDL kódu. Obvod správně pojmenujte v titulku obrázku:

Vše potřebné je v přednáškách.



Dvoubitový registr s asynchronním přednastavením