
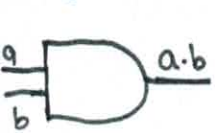


Kontradikce: $f=0$ Tautologie: $f=1 = \bar{a} \cdot \bar{b} + a \cdot b + \bar{a} \cdot b + a \cdot \bar{b}$

Negace NOT: $f = \bar{a} (= \bar{a} \cdot \bar{b} + \bar{a} \cdot b)$ 



Logický součin: AND: $f = a \cdot b$



Logický součet OR: $f = a + b$



$(a \cdot b + \bar{a} \cdot b + a \cdot \bar{b})$



Shefferova fce

NAND: $f = a \uparrow b = \overline{(a \cdot b)} = (\bar{a} \cdot \bar{b} + \bar{a} \cdot b + a \cdot \bar{b})$

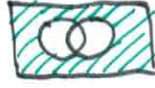


a	b	$a \uparrow b$
0	0	1
0	1	1
1	0	1
1	1	0

a	b	$a \downarrow b$
0	0	1
0	1	0
1	0	0
1	1	0

Pierceova fce

NOR: $f = a \downarrow b = \overline{(a + b)} = (\bar{a} \cdot \bar{b})$



Exkluzivní součet

XOR: $f = a \oplus b = (a \cdot \bar{b} + \bar{a} \cdot b)$




a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Ekvivalence

XNOR: $f = a \equiv b = (\bar{a} \cdot \bar{b} + a \cdot b)$



Identita: $f = a = (a \cdot b + a \cdot \bar{b}) = a \cdot (b + \bar{b}) = a \cdot 1 = a$

Inhibice: $f = a \cdot \bar{b} = (\overline{a \Rightarrow b})$ 

Normální Disjunktivní Forma: termy = IMPLIKANTY = mintermy
 \Rightarrow suma součinů (disjunkce konjunkcí) = SOP = Úplná NDF

$$\begin{aligned} & \bar{x} \cdot \bar{y} \cdot \bar{z} + \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y \cdot \bar{z} + \bar{x} \cdot y \cdot z \\ & \bar{x} \cdot \bar{z} + x \cdot \bar{z} \\ & \bar{z} \end{aligned}$$

Zkrácená NDF = částečně minimalizovaná ÚNDF

Minimální NDF = minimální možné řešení.

Zápis pomocí stav. indexů: $F(x,y,z) = \vee(0,2,4,6) = 1(0,2,4,6) = \sum m(0,2,4,6)$

(stavové indexy, pro které nabývá hod. 1)

Neúplně definováni: $F(x,y,z) = 1(0,2,4) + x(5,6)$

Normální Konjunktivní Forma termy = IMPLICENTY = Maxtermy

\Rightarrow součin sum (konjunkce disjunkcí) = POS = Úplná NKF

$$\begin{aligned} & (x+y+\bar{z}) \cdot (x+\bar{y}+\bar{z}) \cdot (\bar{x}+y+\bar{z}) \cdot (\bar{x}+\bar{y}+z) \\ & (x+\bar{z}) \cdot (x+\bar{z}) \\ & \bar{z} \end{aligned}$$

Sepisujeme hodnoty, ve kterých funkce nabývá 0!

Zápis pomocí stav. indexů: $F(x,y,z) = \wedge(1,3,5,7) = \prod M(1,3,5,7) = \text{TIM}(1,3,5,7)$

Holy grail: $a \cdot b + a \cdot \bar{b} = a$

$(a+b) \cdot (a+\bar{b}) = a$

a DeMorgan se dá zobecnit...

$$\overline{(x+y+z+\dots)} = (\bar{x} \cdot \bar{y} \cdot \bar{z} \cdot \dots)$$

Shefferova algebra: pouze **NAND** $a \cdot b = a \uparrow b$

→ lze v ní realizovat všechny základní členy \Rightarrow všechno

→ komutativní, ne asociativní

$$\overline{a \cdot b} = \overline{b \cdot a} = a \uparrow b = b \uparrow a$$
$$\overline{a \cdot b \cdot c} \neq \overline{a \cdot b \cdot c}$$

$$\overline{a \cdot a} = \overline{a} \quad \overline{a \cdot 0} = 1 \quad \overline{a \cdot 1} = \overline{a}$$

$$\overline{\overline{a \cdot b} \cdot 1} = \overline{\overline{a \cdot b}} = \underline{a \cdot b} \quad \overline{(a \cdot a) \cdot (b \cdot b)} = \overline{a \cdot b} = \underline{a + b}$$

Převody z Booleovy: využití involuce $\overline{\overline{a}} = a$ a De Morgana $\overline{a + b} = \overline{a} \cdot \overline{b}$
(ad democritu)

Pierceva algebra: pouze **NOR** $\overline{a + b} = a \downarrow b$

→ stejně jako Shefferova

$$\overline{a + a} = \overline{a} \quad \overline{a + 0} = \overline{a} \quad \overline{a + 1} = 0 \quad \overline{\overline{a + b} + 0} = \overline{\overline{a + b}} = \underline{a + b} \quad \overline{\overline{a + a} + \overline{b + b}} = \overline{\overline{a + b}} = \underline{a \cdot b}$$

Žegalkina algebra: pouze **XOR** a **AND**

$$a \oplus \overline{a} = 1 \quad a \oplus 1 = \overline{a} \quad a \oplus b \oplus (a \cdot b) = \underline{a + b} \quad 1 \oplus b \oplus (a \cdot b) = a + \overline{b}$$

Rozklady logických fce:

$$\rightarrow \text{v UNDF: } f(a, b, c, d) = \overline{c} \cdot f(a, b, 0, d) + c \cdot f(a, b, 1, d)$$

$$\rightarrow \text{v UNKF: } f(a, b, c, d) = (\overline{c} + f(a, b, 1, d)) \cdot (c + f(a, b, 0, d)) \quad \left. \vphantom{f(a, b, c, d)} \right\} \text{Shannonův teorém}$$

Dualita funkcí: $f(a, b, c, \dots, z, 0, 1, +, \cdot) = f(a, b, c, \dots, z, 1, 0, \cdot, +)$

$$\Rightarrow \text{Rozklad funkce: } F(a, b, c) = ab\overline{c} + \overline{a}b + \overline{a}\overline{b}c = b(a\overline{c} + \overline{a}) + \overline{b}(\overline{a}c)$$

$$\Rightarrow f(b=1) = a\overline{c} + \overline{a} ; f(b=0) = \overline{a}c \quad \text{dá se zapojit multiplexorem}$$

\Rightarrow každá log. funkce se dá implementovat pomocí multiplexorů

Prahová logická funkce

- jednotlivým proměnným přiřazeny váhy i prah T

- prahová funkce = 1, pokud součet hodnot vah proměnných s 1 je $\geq T$

$$f_N^T = 1 \Leftrightarrow \sum_{i=1}^n (w_i \cdot x_i) \geq T \quad \text{řád funkce } N = \sum_{i=1}^n w_i$$

Symetrická prahová log. fce: všechny váhy $w_i = 1$, řád $N =$ počet prom. n

- nabývá hod. 1, pokud alespoň T proměnných = 1

$$F_n^1(a_1, \dots, a_n) = a_1 + \dots + a_n \quad F_n^n(a_1, \dots, a_n) = a_1 \cdot \dots \cdot a_n$$

Majoritní funkce: všechny váhy $w_i = 1$, proměnných je $n \geq 3$

$$\text{- nabývá hod. 1, pokud nadpoloviční většina vstupů = 1} \Rightarrow T = \frac{n+1}{2}$$

Quine - McCluskey pro UNDF (pro UNKF ale vsude hledat na nulý místo jedniček)

$$F(w,x,y,z) = 1(2, 4, 6, 8, 9, 10, 12, 13, 15)$$

1) Tabulka implikanti ve skupinách podle počtu jedniček

skup.1	2	0010	✓
	4	0100	✓
	8	1000	✓
skup.2	2, 6	0110	✓
	8, 9	1001	✓
	8, 12, 10	1010	✓
	8, 4, 12	1100	✓
skup.3	9, 13	1101	✓
skup.4	13, 15	1111	✓

Fajfka znamená, že jsme dvě menší smyčky sloučili do jedné větší

2) Hledáme Booleovské sousední vchody - takové, které se liší jen v jedné pozici

a v sousední skupině

Zkrácené implikanty

2, 6	0-10	PI2
2, 10	-010	PI3
4, 6	01-0	PI4
4, 12	-100	PI5
8, 9	100-	✓
8, 10	10-0	PI6
8, 12	1-00	✓
8, 12, 9, 13	1-01	✓
8, 9, 12, 13	110-	✓
13, 15	11-1	PI7

tyhle už žádné sousední termíny nemají!

stejný řádek!

Do nové tabulky zapsat jen jednou!
Další už ale dělat nebudu.

→ 8, 9, 12, 13 | 1-0- PI1

3) Opakujeme pro další dvojice, které se liší jen v jedné pozici.

→ Smyčky bez fajfky = množina smyček/minimálních implikanti, ze kterých budeme hledat řešení.

⇒ Nějak si je označ! PI1, 2, 3...

4) Hledáme kombinaci smyček, aby řešení bylo minimální: vytvoříme mřížku implikanti

	2	4	6	8	9	10	12	13	15
PI1				X	X		X	X	
PI2	X		X						
PI3	X					X			
PI4		X	X						
PI5		X					X		
PI6				X		X			
PI7								X	X

1) Sloupce pokrývá pouze jedním implikantem: 9, 15
⇒ PI1 + PI7 tam budou!
⇒ pokrýj i další: 8, 12, 10

2) Hledáme kombinaci, které nejefektivněji pokryje zbytek
⇒ PI3 + PI4
(ad dále)

nesporné implikanty

5) Nejlepší řešení je tedy

$$F(w, x, y, z) = \text{PI1} + \text{PI3} + \text{PI4} + \text{PI7}$$

$$= (1-0-0) + (-010) + (01-0) + (11-1)$$

$$= \underline{\underline{w\bar{y} + \bar{x}y\bar{z} + \bar{w}xz + wxz}}$$

Pro X vrcholy: zařadit do tabulky, jako by byly v log. 1,
ale v kroku 4 nepoužívat v soupisu + ignorovat smyčky, kde jsou jen ony

Petrická funkce:

1) Přepsat tabulku bez nesporných implikanti:

	2	4	6	10
PI2	X		X	
PI3	X			X
PI4		X	X	
PI5		X		
PI6				X

2) Ze sloupců konjunkce +, zapset jako
souvětín → vznikne NKF.

(Říká: „Vrchol 2 můžeme pokrýt PI2 NEBO PI3.“)

$$(PI2 + PI3) \cdot (PI4 + PI5) \cdot (PI2 + PI4) + (PI3 + PI6)$$

3) Upravím na NDF.

$$= PI2 \cdot PI3 \cdot PI5 + \text{PI3} \cdot \text{PI4} + PI2 \cdot PI4 \cdot PI6 + PI2 \cdot PI5 + PI6$$

4) Vyberu pokrýt s nejmenším počtem implikanti.

Kombinační obvody = "krabičky se vstupy a výstupy, které pouze provádějí nějakou fci"

- dá se popsat tabulkou (pravdivostní)
- bez cyklů / zpětných vazeb! (to jsou sekvenční obv.)
- seskupují se do funkčních modulů - typicky (de)multiplexor, (de)kodér, sčítáčka, násobička ...

Demultiplexor = posílá 1 vstup na jeden z výstupů podle zvolené adresy

- 1 vstup D
- N adresových vstupů $A_{N-1} \dots A_0$
- 2^N výstupů $Y_{2^N-1} \dots Y_0$

DMX $1-2^N$

$$Y_i = D \cdot m_i$$

\hookrightarrow minterm i určený adr. A

(- občas se nahrazuje až uvnitř jednotliv. log. obvodů \rightarrow všechny mají připojené D a kontrolují si adresu)



$$Y_0 = D \cdot m_0 = D \cdot \bar{A}_1 \cdot \bar{A}_0$$

$$Y_1 = D \cdot m_1 = D \cdot \bar{A}_1 \cdot A_0$$

$$Y_2 = D \cdot m_2 = D \cdot A_1 \cdot \bar{A}_0$$

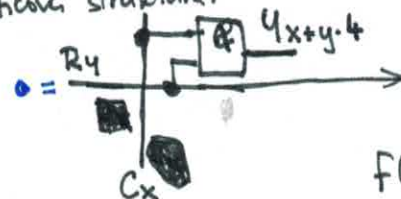
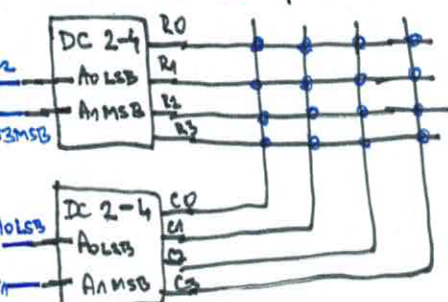
$$Y_3 = D \cdot m_3 = D \cdot A_1 \cdot A_0$$

4 AND hradla (3vstupá)
+ 2 NOT hradla

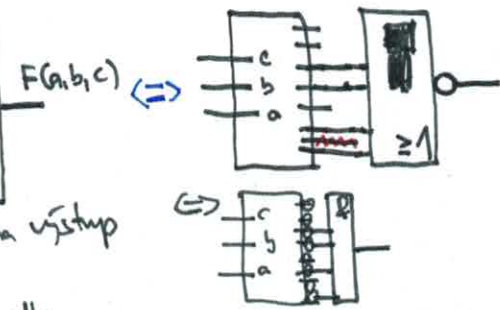
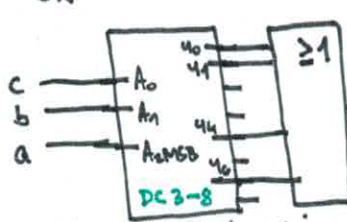
Dekodér = demultiplexor s $D=1$ fixně \rightarrow převádí "adresu" na "číslo" **DC $N-2^N$ ($1-2^N$)**

\hookrightarrow binární dekodér

Dekodér 4-16: používá se maticová struktura:



$$F(a,b,c) = \sum m(0,1,4,6) = \Pi M(2,3,5,7)$$



použití pro realizaci lib. log. fce:

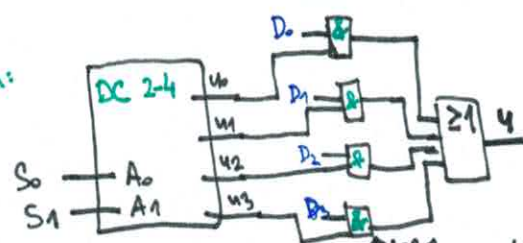
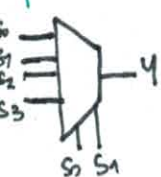
Multiplexor - podle adresy vybere 1 ze vstupů a pošle jej na výstup

- N adresových vstupů $S_{N-1} \dots S_0$ (S =selektor)
- 2^N datových vstupů $D_0 \dots D_{2^N-1}$
- 1 výstup Y

MX 2^N-1

$$Y = \sum_{i=0}^{2^N-1} (m_i \cdot D_i)$$

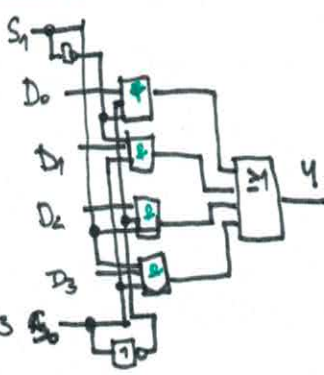
pomocí dekodérů:



běžně:

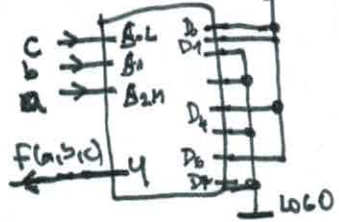
MX 4-1:

$$Y = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3$$



pro realizaci lib. log. fce:

$$F(a,b,c) = \sum m(0,1,4,6)$$



nepotřebujeme hradlo navíc jako u dekodéru \heartsuit
pokud chceme fci 4 prom., ale máme jen 8-1 mult.
podíváme se, jak se fce chová vůči zbývajícím proměnným
Pokud je na ní závislá, napojíme do příslušného D té proměnné (nebo negaci)

$$f(a,b,c) = ab + \bar{b}c = ab(c + \bar{c}) + (a + \bar{a})\bar{b}c$$

$$= \underline{abc} + \underline{ab\bar{c}} + \underline{a\bar{b}c} + \underline{\bar{a}\bar{b}c} \Rightarrow ab$$

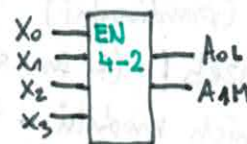
$00 \text{ } a\bar{b} \Rightarrow D_0 = 0$
 $01 \text{ } a\bar{b} \Rightarrow D_1 = 0$
 $10 \text{ } a\bar{b} \Rightarrow D_2 = 0$
 $11 \text{ } a\bar{b} \Rightarrow D_3 = 0$

Kódér - převádí kód z 2^N na N výstupů ("čísla" na "adresu/binární číslo")

- N adresových výstupů $A_0 \dots A_{N-1}$

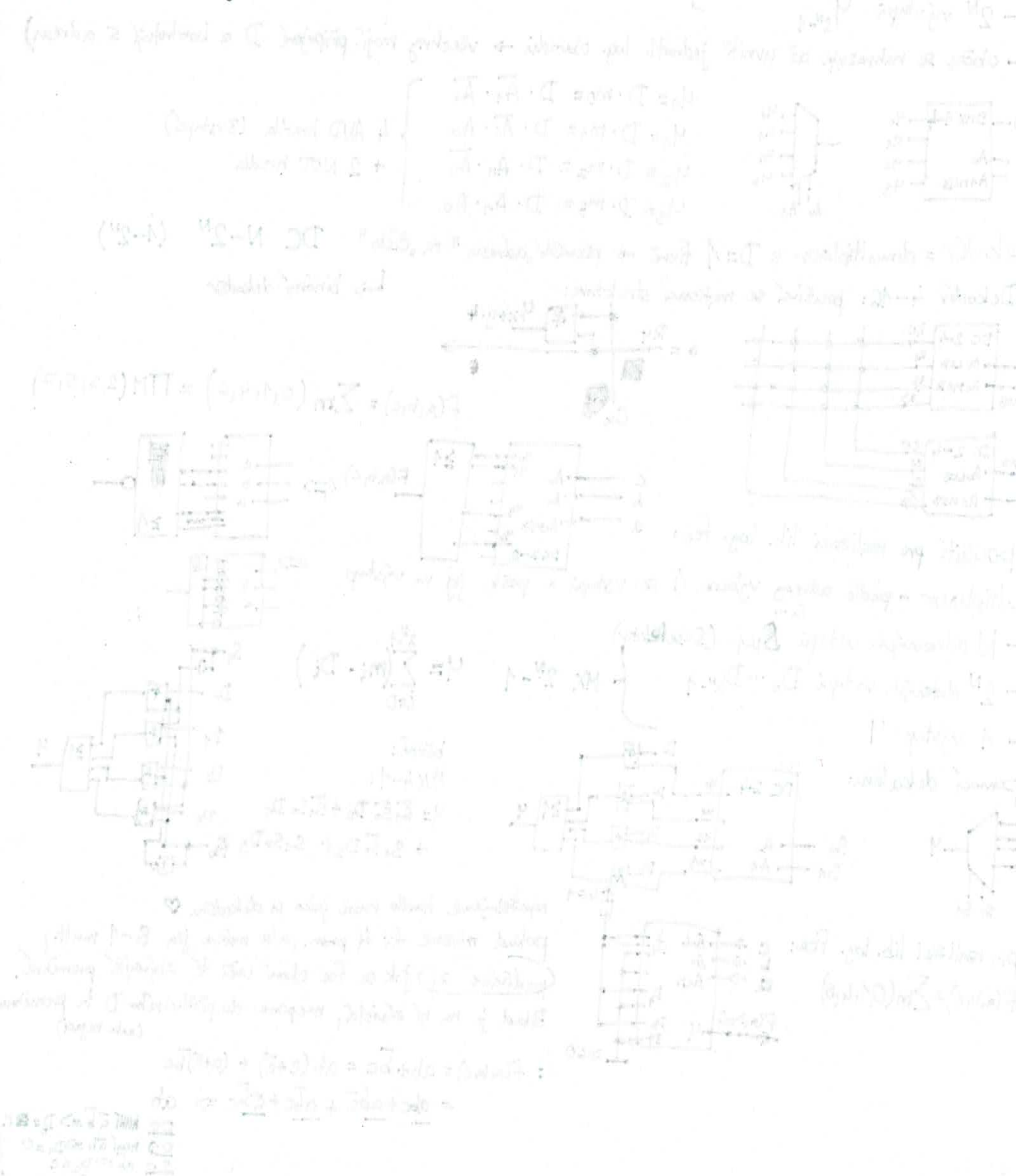
- 2^N vstupů $X_0 \dots X_{2^N-1}$

} EN 2^N-N



- problém: v základní variantě může být aktivní pouze jeden vstup X ! (hodně výstupních stavů nepovolen!) \Rightarrow ak jednoduché realizace X_1 X_2 X_3 X_4 A_0 A_1

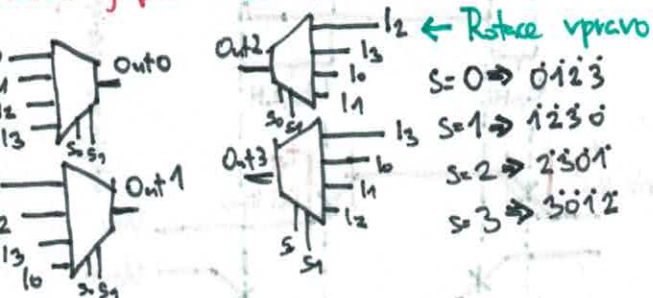
\rightarrow Prioritní kódér - určuje, který nejvyšší vstup je aktivní + výstup GS - detekce, že nějaký aktivní je.



Test všech = 1: prostě AND Test všech = 0: prostě NOR $a_0 + a_1 + \dots + a_n$

Test shody dvou n -bitových vektorů: $(a_0 \oplus b_0) + (a_1 \oplus b_1) + \dots + (a_{n-1} \oplus b_{n-1})$ XOR určuje nerovnost
 \Rightarrow výstupy všech XORů musí být $= 0$, pokud aspoň jeden $= 1 \Rightarrow$ nerovnost

Valcový posuvac (barrel shifter):



- zbytečně složitý

- hodně moc vodičů - 64×64 + výstupy + selektory...

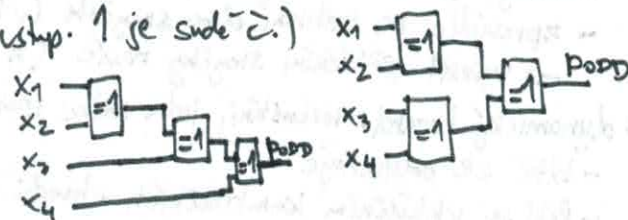
\Rightarrow typický v CPU jen posuvac o jeden bit

Comparator: obvykle realizace odčítáním (sčítáčka a opačného č.), lze i kombinačním obvodem:
 \rightarrow vyrobiť pravdivostní tabulku, minimalizovat...

Parita: lichá (počet vstupních 1 je lichý č.) x sudá (počet vstup. 1 je sudý č.)

\rightarrow jednoduše XORy: $Parity = x_1 \oplus x_2 \oplus x_3 \oplus \dots \oplus x_n$

- stromem nebo kaskádou



Dvořiční binární sčítáčka (HA - half adder)

- 2 (jednotlivé) vstupy x, y

- výstup S

- výstup Carry

- nemá vstup carry - proto poloviční

$$S = x \oplus y$$

$$C = x \cdot y$$

Úplná bin. sčítáčka (FA)

- 2 (b) vstupy x_i, y_i

- vstup C_{i-1} - carry in

- výstup S_i

- výstup C_i

$$S_i = y_i \oplus x_i \oplus C_{i-1}$$

$$C_i = x_i \cdot y_i + x_i \cdot C_{i-1} + y_i \cdot C_{i-1}$$

$\Rightarrow = 1$, pokud aspoň 2 ze 3 vstupů $= 1$

Vícebitové sčítáčky:

- možnost kaskádou (za sebou) zapojit N jednotliv. FA = ripple carry (jako na papíře pod sebou)

- pomalé! \Rightarrow různé rychlejší varianty - carry look-ahead (není sekvenční)

Detekce přetečení (v doplňkovém kódu):

a) přenos C ze znaménkového bitu \Rightarrow přenos C do znaménkového bitu ($O_f = (C_f \neq P_f)$)

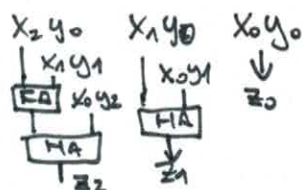
b) porovnání znaménkových bitů sčítanců a součtu

Vásobička bez znaménka:

a) řetěz FA/HA 1bit sčítáček

- pomalé

b) stromová struktura



Vásobička se znaménkem:

a) kladný násobitel (spodní) \Rightarrow je třeba šířit znaménko v mezinásobcích

b) záporný násobitel \Rightarrow b.i) obětit znaménka násobence i násobitele

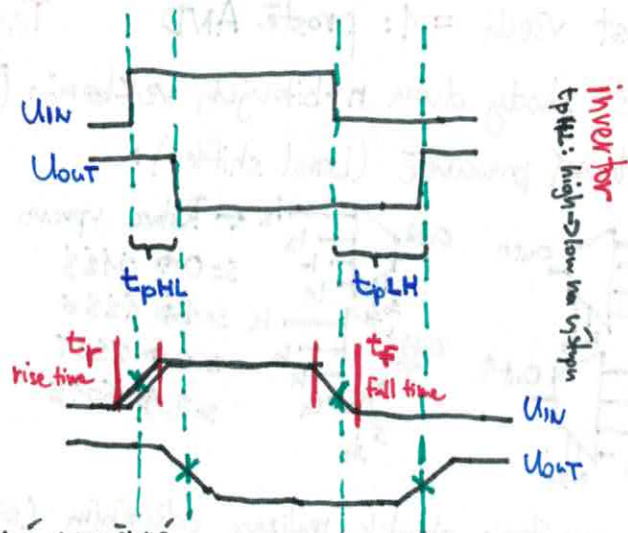
b.ii) při násobení znaménkovým MSb korekce přičtením dvojkového doplňku násobence

Chování v čase

- každý obvod má zpoždění t_d - nejhorší potřebná doba
- často závisí na prostředí - teplotě
- problém: neostrost reálných průběhů - rising & falling time
 - aproximace lichoběžníkem
 - šířka pulzu: podle středních hodnot
 - nízký čas pro přechod $0 \rightarrow 1$ a $1 \rightarrow 0$
- setrvačné zpoždění: příliš krátké pulzy neprojdou (inertní zp.)
- transportní zpoždění: rychlost šíření signálu v médiu

Hazardy: při probublávání se výstupy rozkmitávají

- **statický hazard:** změna jedné proměnné → přechodná změna jiné proměnné
 - zpravidla na rozhraní dvou smyček (v K mapě)
 - vyřešit přidáním smyčky navíc (X už není minimální)
- **dynamický hazard:** rozkmitání, jedna změna ($0 \rightarrow 1$) udělá třeba ($1 \rightarrow 0 \rightarrow 1 \rightarrow 0$) místo ($1 \rightarrow 0$)
 - bře se odstraní
 - řeší se vkládáním kombinačních obvodů do sekvencí (s clock signálem)



Sequenční obvod = kombinální obvod + paměť, výstup závisí na současných i minulých hodnotách

- vektor vnitřního/současného stavu $Q (=q_i)$ představuje PS
- přechodová funkce $\rightarrow P (=q_{i+1})$ next state NS
- paměť **volatívní** (kdykoliv R/W) nebo **nevolatívní** (kdykoliv R, přepset se ale musí celá)
- jako základní paměťové prvky se používají **klopné obvody** - zpoždění v nich

Modelování: $KA = (X, Y, Q, q_0, P, V)$

X : vstupní abeceda (vstup. proměnné)

Y : výstupní abeceda (výstup. p.)

Q : vnitřní abeceda

$q_0 \in Q$: počáteční stav

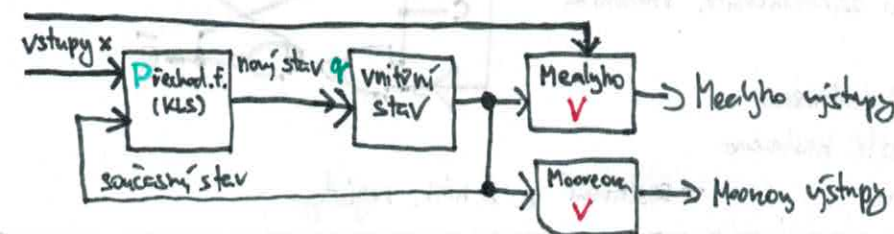
P : přechodová funkce

V : výstupní funkce

$P: X \times Q \rightarrow Q, q_{i+1} = P(x_i, q_i)$

Mealyho KA: výstup je funkcí stavu i vstupu: $V = X \times Q \rightarrow Y$ $y_i = V(x_i, q_i)$

Mooreův KA: výstup je funkcí pouze stavu: $V = Q \rightarrow Y$ $y_i = V(q_i)$

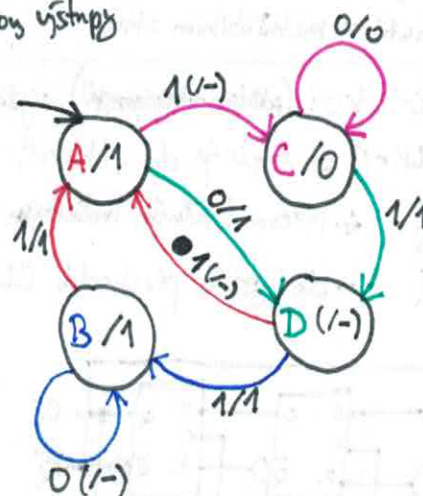


Moore:

q_{i+1}	X	
q_i	0	1
A/1	D	C
B/1	B	A
C/0	C	D
D/-	A	B

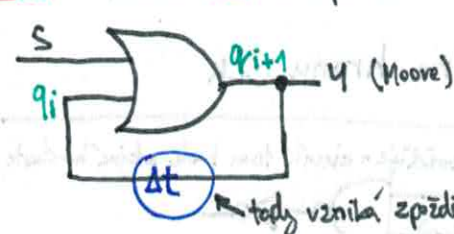
Mealy:

q_{i+1}	X	
q_i	0	1
A	D/1	C/-
B	B/-	A/1
C	C/0	D/1
D	A/-	B/1

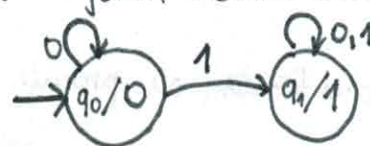


nedefinované výstupy není třeba uvádět

SET: hradlo OR se zpětnou vazbou:

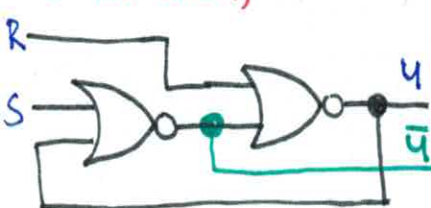


tedy vzniká zpoždění



\rightarrow není to moc užitečné
 \rightarrow jak přidat reset?

R-S KO (latch) - překlápí se mezi stavy na vstupu R, S - hlediny \rightarrow hledinový KO $Q_{i+1} = S + \bar{R} \cdot Q_i$

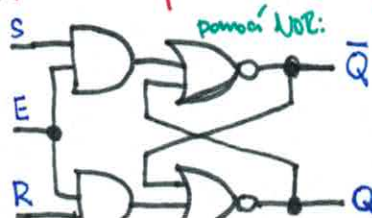


- má dva stabilní stavy \rightarrow **bistabilní**

- při přechodu se přelévá napětí a nějakou dobu je obvod v **metastabilním** s neplatným stavem: $R=1, S=1$ (nežádoucí chování po uvedení zpět do povelového st.)

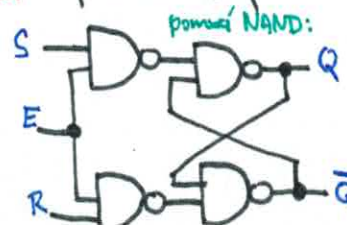
- výstupy Q a \bar{Q} nejsou v době přechodu komplementární

R-S KO s povelovým vstupem



pomocí NOR:

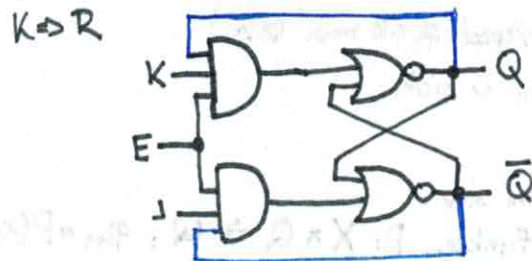
- přidává vstup E (/C-control) - funguje jako HOLD



pomocí NAND:

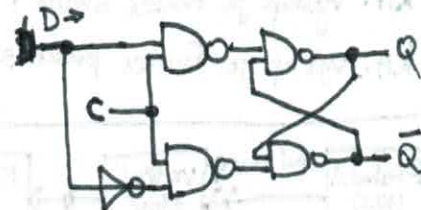
$$Q_{i+1} = S \cdot E + (\bar{R} + E) \cdot Q_i$$

J-K KO s E - řeší problém nedefinovaného stavu 11
 „jedničky“ $\Rightarrow S \rightarrow$ v kombinaci 11 dělá toggle - překlápí do opačné hod.



\rightarrow **T KO (s E)** - vznikne spojením K a J \Rightarrow vstup Toggle
 - pokud $E=1 \wedge T=1$, výstup začne oscilovat \Rightarrow je třeba zavést hodinový signál

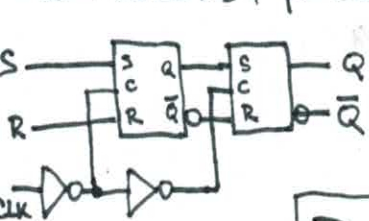
D KO - ukládá hodnotu vstupu D, pouze pokud je C aktivní
 - pokud je C neaktivní, výstupem je poslední zaznamenaná hodnota
 - v podstatě RS s $S=\bar{R}$
 - **setup time**: kolik času před C musí už být D nastaveno
 - **hold time**: kolik času po C musí být D ještě nastaveno
 \Rightarrow nedodržení \Rightarrow zaseknutí v metastabilním stavu



- sestávají se z nich registry

Dvoufázový (Flip-Flop) R-S ko: (edge-triggered), vyžaduje hodinový signál CLK

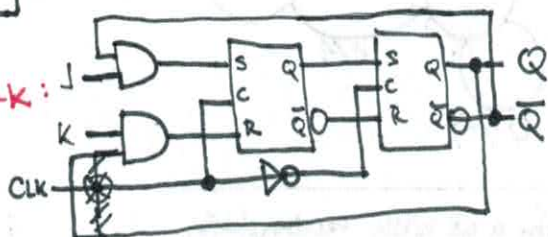
- dva RS KO s E, při CLK=0 se zapisuje do „Master“, při CLK=1 se přepíše do „Slave“



\leftarrow řízeno kladným hodinovým pulzem, značka \downarrow

- do grafu přechodů CLK napíšeme!

Dvoufázový J-K:



\leftarrow řízeno záp. pulzem, \uparrow

Dvoufázový T KO - spojení J a K...

Derivační D KO - funkce stejná jako u dvoufázových, ale pracuje jen na hranu CLK

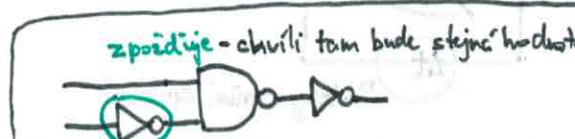
- téměř pouze v diskretním čase
 = synchronní edge-triggered

Jak detekovat náběžnou hranu? - vytvořením statického hazardu

- do jedné větve zavedeme invertor, který způsobí zpoždění \rightarrow na výstupu krátký pulz odpovídající zpoždění

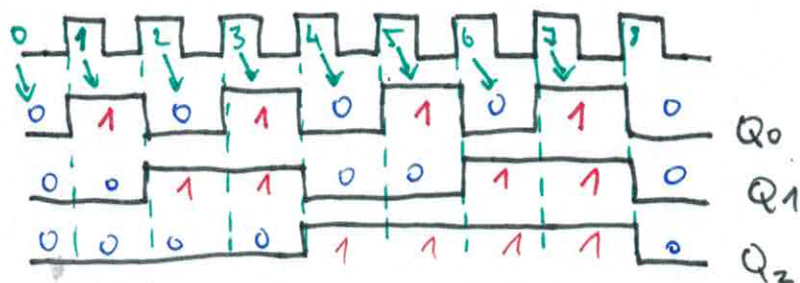
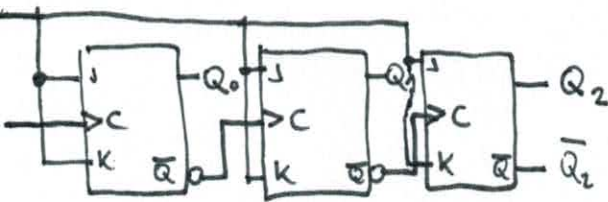
Povolování hodinového sig: - hradlem \overline{CE} ne - vytvoří zpoždění

- sněžíme se naposled CE někde na datové cestě



- VHDL** - programovací jazyk pro popis čís. obvodů (HDL = hardware description lang.)
- umožňuje jednoznačně modelovat a simulovat
 - syntéza - transformuje HDL popis do prvků cílové technologie
- Komponenta** - popisuje HW prvek
- vstupy, výstupy, vnitřní logika
- **Entita** - rozhraní mezi komponentou a okolím
- signály (I/O): sekce **port**
 - generické parametry (např. datová šířka)
- **Architektura** - tři popisy: **strukturální** - z jakých komp. se skládá
behaviorální - programování, co se stane, když...
data flow - datový tok, jak data plynou

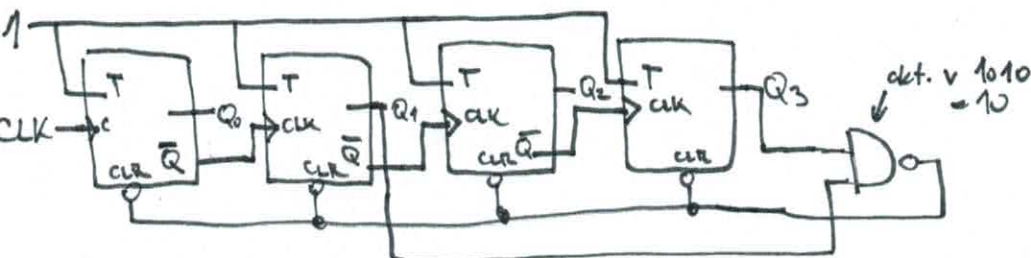
Asynchronní čítač:



- nevýhoda: trvá, než se ustálí hodnota na výstupu (musí to probublat)

Asynchronní čítač mod 10:

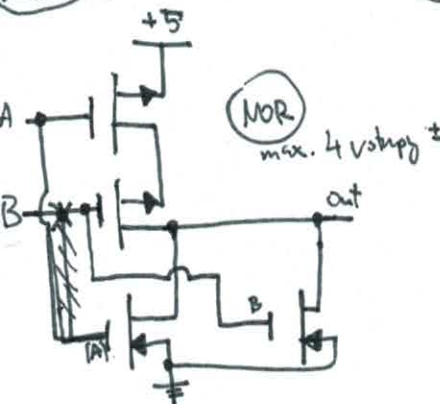
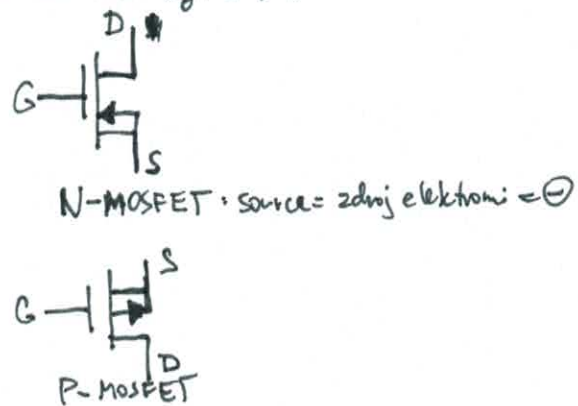
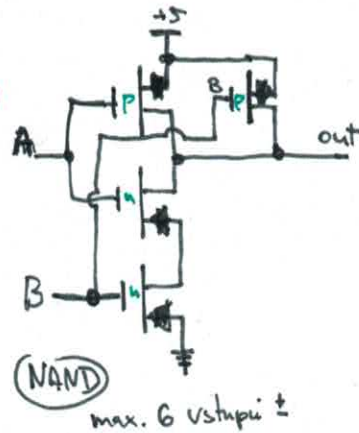
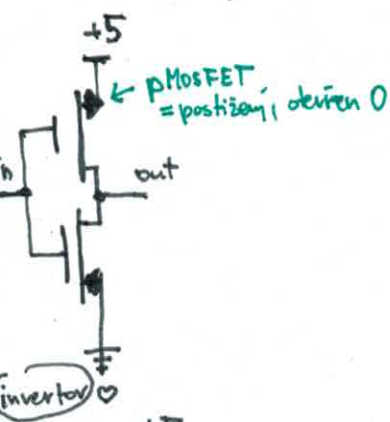
- dekodér stavu 10, který asynchronně nuluje všechny KO → přivádí všech hodnot na CLR



- chvíli tam bude ta hodnota 10 = **glitch** → je třeba s tím počítat!

CMOS - komplementární MOS

- použití unipolárních MOSTETŮ \Rightarrow prakticky netečou proudy
- velmi velký odpor OFF, velmi malý odpor ON \Rightarrow velmi kvalitní log. 0 i 1



- hystereze: chová se to jinak v $0 \rightarrow 1$ než $1 \rightarrow 0$
- teplota - cíl 22° , zapne až nad $22,5^\circ$, vypne pod $21,5^\circ$

SRAM: buňky v podstatě R/S KO, nějaký SELECT vstup a dva invertory do kříže

DRAM: hodnota se uchovává jako náboj - parazitní kapacita - musí se pravidelně osvěžovat

- jeden velmi malý tranzistor

Funkční generátor: funguje jako paměť, posuvný registr nebo LUT (lookup table)

SLICE: fun. gen., registry, MUXFx, pomocná logika - Carry, aritmetika

LUT: základní log. hradlo

N-bitový vstup, 1bitový výstup (Spartan: $N=4$)

libovolná binární fce. N proměnných

MUXFx: multiplexory umožňující realizaci funkce více proměnných x = bit, který to přepíná

RAM v FG: 2^N bitů, asynchronní čtení, synchronní zápis (write enable, write clock)