Úvod do počítačových architektur

T.Mainzer

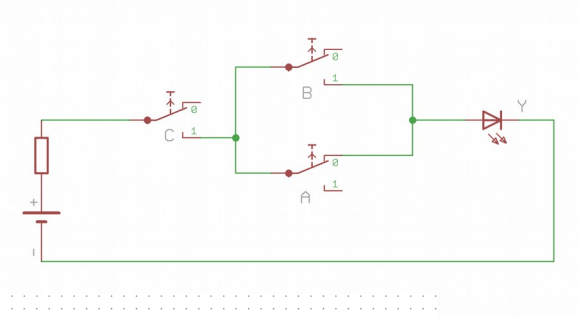
Úvod

- analogový vs digitální počítač
- analogový - (+) rychlost, (-) přesnost, opakovatelnost, specializovanost
- digitální - (+) opakovatelnost, univerzálnost

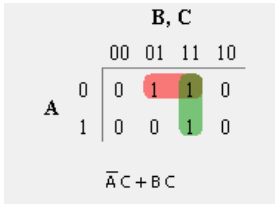
Kombinační logické (digitální) obvody

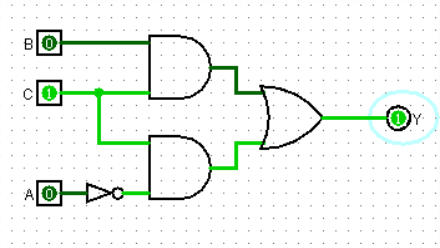
- 0/1
- výstup je fcí vstupu, tj. bez paměti (vs. sekvenční obvody – s pamětí)
- závislost vstupu a výstupu lze popsat
- - výraz (logická funkce)
- - tabulka
- - Karnoughova mapa (K-mapa)
- - log.obvod
- - (schéma, diagram, program (Verilog, VHDL))

Příklad1 (vše popisuje jednu fci) – je vzájemně převoditelné

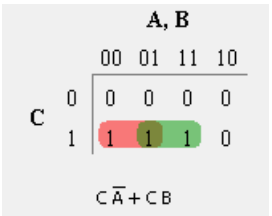


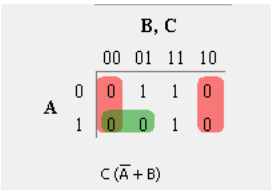
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

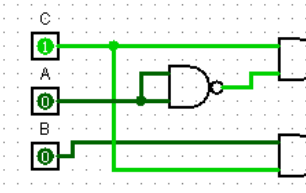


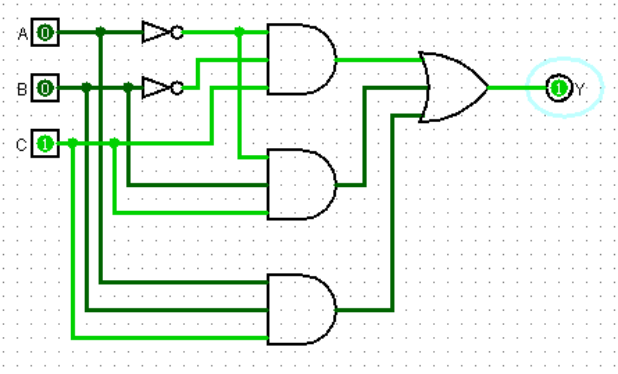


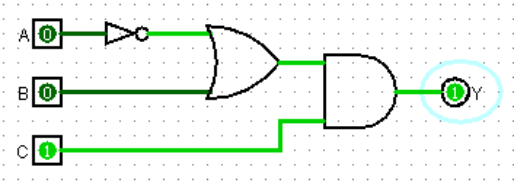
Y = B.C + ~A.C
Y = ~((~(B.C)) . (~(~A.C)))
Y = ~A.~B.C+~A.B.C+A.B.C
Y = C.(B+~A)











Booleovská algebra

!x, -x, ~x, $\bar{x}$ = negace,inverze (~0=1, ~1=0)

x.y, x&y = log.soucin, and (0&0=0, 0&1=0, 1&0=0, 1&1=1)

x+y, x|y = log.součet, or (0+0=0, 0+1=1, 1+0=1, 1+1=1)

komutativita: x + y = y + x, x.y = y.x

distributivita: x + (y.z) = (x+y) . (x+z), x. (y+z) = (x.y) + (x.z)

neutralita 0/1: x+0 = x, x.1 = x

komplementarita: x + ~x = 1, x.~x = 0

(!pozor, + a . je jen symbol log.fcí OR a AND, není to arimetické + a *)

- asociativita: x + (y+z) = (x + y) + z, x.(y.z) = (x.y).z

- absorpce: x + (*x.y*) = *x*, *x.(x+y)* = *x*

- agresivita nuly,jedničky: x.0 = 0, x+1 = 1

- idempotence: x+x = x, x.x = x

- absorce negace: x+(~x.y) = x+y, x.(~x+y) = x.y

- dvojitá negace: ~(~x) = x

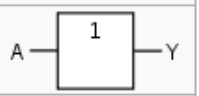
- komplementarita 0/1: ~0=1, ~1=0

- DeMorganovy zákony: ~x.~y = ~(x+y), ~x+~y = ~(x.y)

- DeMorganovy zákony: x.y = ~(~x+~y), x+y = ~(~x.~y)

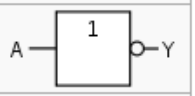
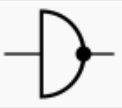
Základní logické funkce (hradla)

Opakovač, Repeater

Funkce	$\mathbf{Y = A}$	
Značení		Pravdivostní tabulka
norma	symbol	
ANSI/MIL		
IEC		
DIN		

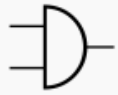
$X(A)$	Y
0	0
1	1

NOT, Invertor

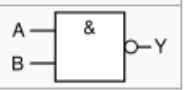
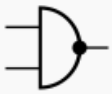
Funkce	$\mathbf{Y = \bar{A}}$	
Značení		Pravdivostní tabulka
norma	symbol	
ANSI/MIL		
IEC		
DIN		

$X(A)$	Y
0	1
1	0

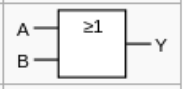
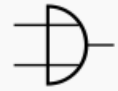
AND, log.soucin, konjunkce

Funkce	$Y = A \cdot B$		
Značení		Pravdivostní tabulka	
norma	symbol		
ANSI/MIL		$X_1(A)$	$X_2(B)$	Y
IEC		0	0	0
DIN		0	1	0
		1	0	0
		1	1	1

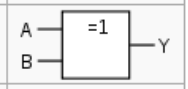
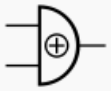
NAND, negovaný log.součin

Funkce	$Y = \overline{A \cdot B} = \overline{A} + \overline{B}$		
Značení		Pravdivostní tabulka	
norma	symbol		
ANSI/MIL		$X_1(A)$	$X_2(B)$	Y
IEC		0	0	1
DIN		0	1	1
		1	0	1
		1	1	0

OR, log.součet, disjukce

Funkce	$Y = A + B$		
Značení		Pravdivostní tabulka	
norma	symbol		
ANSI/MIL		$X_1(A)$	$X_2(B)$	Y
IEC		0	0	0
DIN		0	1	1
		1	0	1
		1	1	1

XOR, exkluzivní log.součet

Funkce	$Y = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$		
Značení		Pravdivostní tabulka	
norma	symbol		
ANSI/MIL		$X_1(A)$	$X_2(B)$	Y
IEC		0	0	0
DIN		0	1	1
		1	0	1
		1	1	0

Program Logisim

Program pro simulaci dig.log.obvodů

http://www.cburch.com/logisim/, https://sourceforge.net/projects/circuit/

Poznámky k návrhu

schéma, rovnice - plně definované (bez nedefinovaných stavů), dosazením hodnot možno vytvořit tabulku a mapu.

tabulka, karnaughova mapa – může obsahovat nedefinované hodnoty (prázdné údaje) – ty můžeme dodefinovat abychom dosáhli výslednou co nejjednodušší implementaci (alternativně tím zjistíme že úloha není plně definovaná). Z řádků tabulky lze přímo psát rovnici ve tvaru součet součinů, ale není optimalizovaná.

zjednoduššování, optimalizace – úpravou rovnice (booleovská algebra), minimalizací v karnaoughově mapě

návrhová pravidla mají přesah i do SW – kontrola úplnosti/korektnosti zadání, zjednodušování či návrh podmínkových pravidel (IF ...)

Příklad1b – modifikace/pokračovavání, nedefinované stavy

Zadání – jsou-li vsechny spínace A,B,C seplé(=1) má dioda svítít. Je li B a C rozepnuté (=0) nemá svítít. Je li B a C seplé má svítit. Je li A zaplé a B rozeplé nemá svítít. Je li A a C zaplé a B rozeplé nemá svítit.

Hloupý programátor / návrhář	Běžný programátor	Všímavý programátor	Programátor znající minimalizaci

Boolean y;
if ((a==1)&(b==1)&(c==1)) y=1;
if ((b==0)&(c==0)) y=0;
if ((b==1)&(c==1)) y=1;
if ((a==1)&(b==0)) y=0;
if ((a==1)&(b==0)&(c==1)) y=0;
	Boolean y=0;
	if ((a==1)&(b==1)&(c==1)) y=1;
	if ((b==1)&(c==1)) y=1;
		Boolean y=0;
		if ((b==1)&(c==1)) y=1;
			Boolean y=b;

a	b	c	y
0	0	0	0
0	0	1	x
0	1	0	x
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	x
1	1	1	1

	b, c			
	00	01	11	10
a 0	0	x	1	x
a 1	0	0	1	x

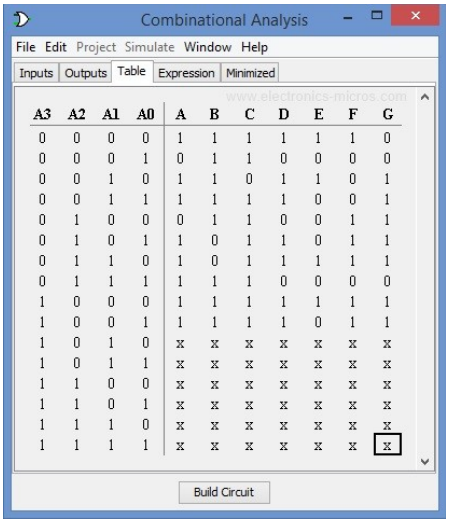
Příklad2 – koder(-dekoder) (více výstupů, nedefinovené stavy)

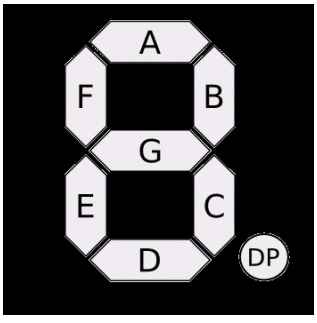
Koder AB→WXYZ, dekoder WXYZ→AB (adresový dekoder, chipselect)

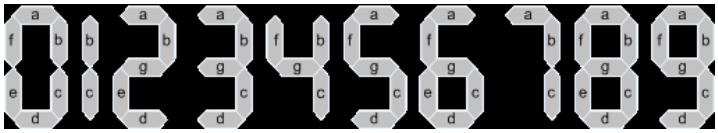
Koder AB<->WXYZ					
A	B	W	X	Y	Z
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

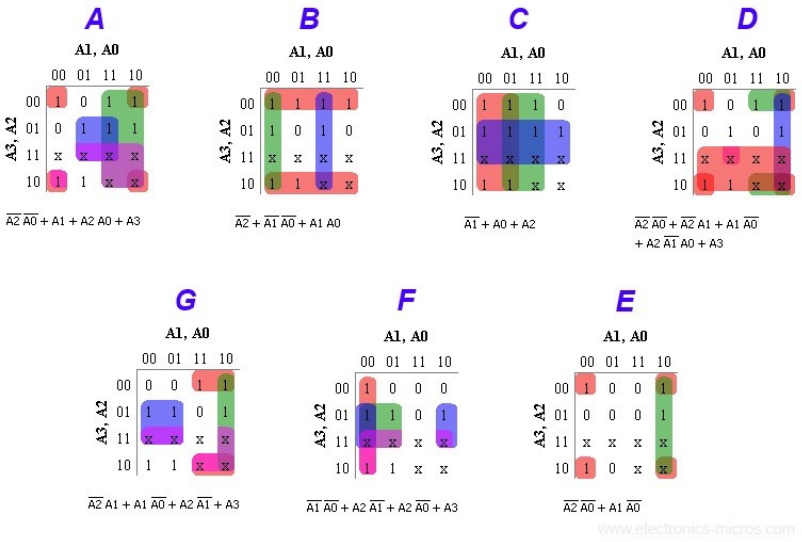
Příklad3 – dekoder na sedmisegmentovku

Koder A3A2A1A0 -> ABCDEFGH









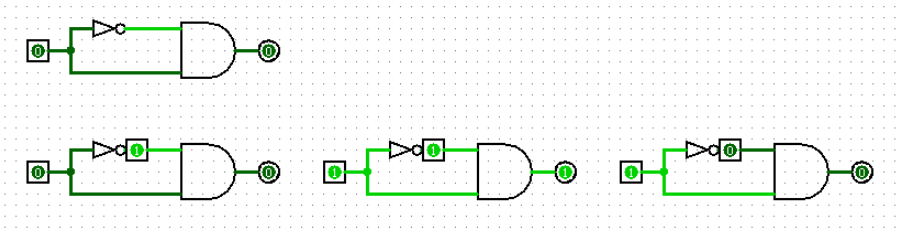
upozornění na chybu: růžová barva v K-diagramech není zobrazena celá (digragr.A,D,G,F)

pozn: možné použití paměti ROM (ve smyslu kombinačního obvodu adresa->data)

Příklad4 – Hazard

Reálné obvody – konečná doba šíření signálu, zpoždění na hradlech

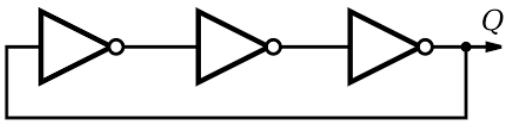
Příklad: Y = ~A&A = 0



Ale: (reálné) hradlo negace má (také) zpoždění a při změně vstupu A $0 \rightarrow 1$ (viz prostřední obr kdy signál ještě neprošel hradlem) bude výstupní hodnota $0 \rightarrow 1 \rightarrow 0$. Tj. Ustálený stav OK, ale přechodně je tam nesprávná hodnota (=hazard)

Příklad5 – asynchronní sekvenční obvod

(ring loop):



Sekvenční logické obvody

Ke kombinační části se přidává paměťová, tj. Výstup je fcí vstupů + vnitřních proměných (minulost). tj. stejné vstupy nemusejí vyvolávat stejné vystupy, ale jsou závislé na vnitřním stavu (paměti)

Sekvenční obvody dělíme na

- **asynchronní** – změna vstupů se ihned promítne do stavu sekvenčního obvodu – (+) rychlost (-) složitost návrhu (hazardy)

- **synchronní** – zde je zaveden řídicí synchronizační signál (hodinový signál, hodiny). Změna vstupní proměnné se promítne do stavu sekvenčního obvodu až při příchodu hodinového signálu.

Dle reakce na hodinový signál můžeme dále rozlišit

- - obvod řízený hladinou hodinového pulsu (0, 1)

- - obvod řízený hranou hodinového pulsu (vzestupnou 0->1, sestupnou 1->0, oběma)

- **Moore / Mealy –** u Moorova je vystup je funkcí stavu, u Mealyho je fcí stavu+vstupů

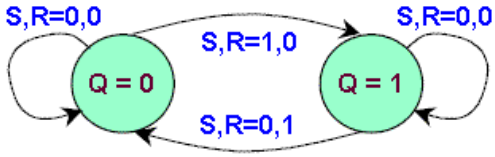
Sekvenční log. Obvod lze (navíc ještě) popsat:

- stavovým diagramem (stavový automat)

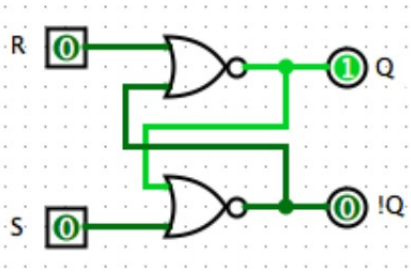
- sekvenčním diagramem (průběhy signálů)

Pamětové členy, klopné obvody flipflop

RS klopný obvod (reset/set)

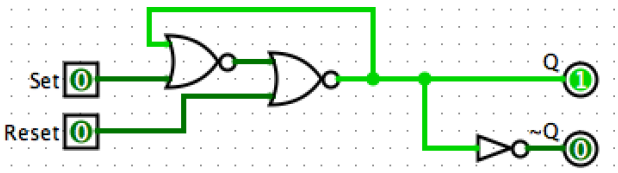


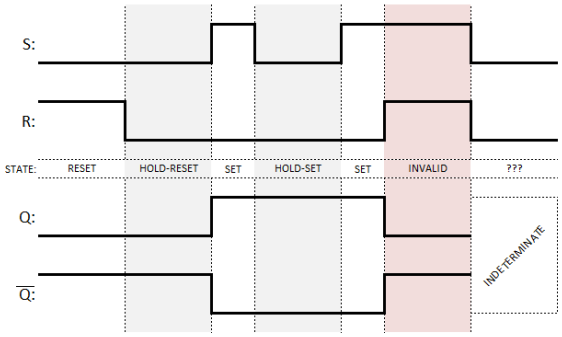
Asynchonní (zpětná vazba), NAND – paměťový prvek, pro RS=00 uchovává informaci



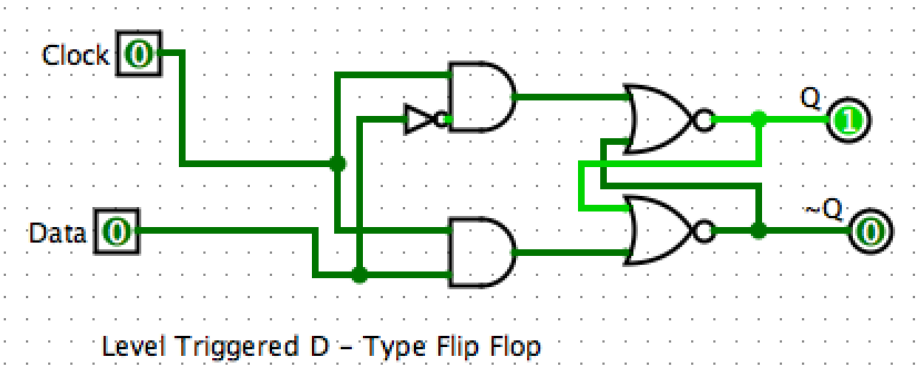
R	S	Q	~Q	pozn
0	0	Q	~Q	Beze změny
0	1	1	0	
1	0	0	1	
1	1	0	0	„Zakázaný“ stav

RS Při „standardním“ návrhu:



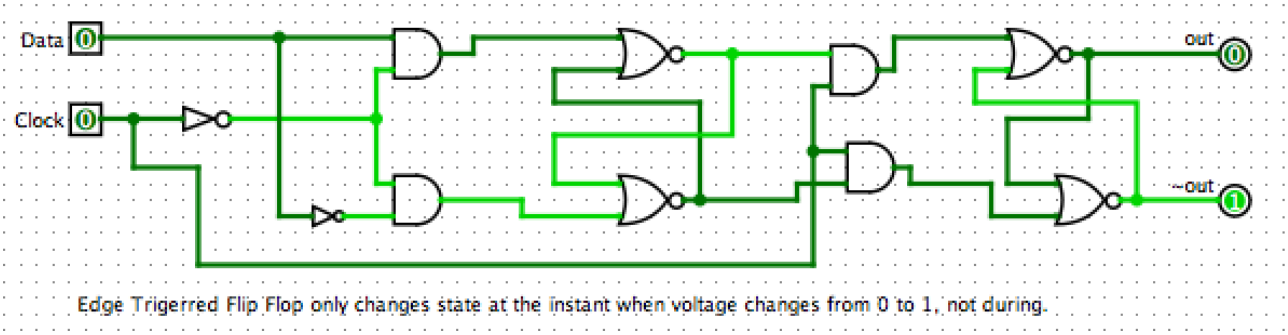


D klopný obvod (data)



Clock	**Data**		**Q**	**~Q**	**pozn**
0	X		Q	~Q	Beze změny
1	D		D	~D	

Je li clock==0 uchovává data poslední známá data(D), je li Clock==1 je možná změna informace (=obvod řízený hladinou)



Uchovává data jen v okamžiku změny clock 0→1 (=obvod řízený hranou)

pozn ke schématu: je to 2*D klopný obvod za sebou, kde první má negovaný signál clock.

Clock	**Data**		**Q(t+1)**	**~Q(t+1)**	**pozn**
X	X		Q	~Q	Beze změny – obvykle se neuvádí
0->1	D		D	~D	

T klopný obvod (trigger)

Clock	**Data**		**Q(t+1)**		**pozn**
X	X		Q		
0->1	0		Q		Beze změny
0->1	1		~Q		Negace

JK klopný obvod (Jordan-Eccles)

Popis chování:

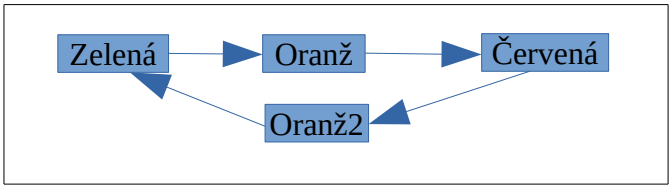
Clock	**JK**		**Q(t+1)**		**pozn**
X	XX		Q		Beze změny (reaguje jen na hranu hodin)
0->1	00		Q		Beze změny
0->1	01		0		
0->1	10		1		
0->1	11		~Q		Negace

Popis chování ve formě vhodné pro návrh sekvenčních obvodů

JK	**Q(t)**		**Q(t+1)**		**pozn**
0X	0		0		
1X	0		1		
X1	1		0		
X0	1		1		

Příklad1 – semafor

Zelená pro auta. Zmáčkne li chodec tlačítko dostanou auta na chvíli červenou.



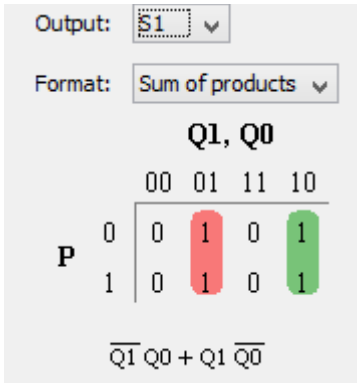
Stav (Q1Q0)	**Vstup (P)**		**Nový stav (S1S0)**	**Výstup (ZeOrČe)**
Zelená (00)	0		Zelená (00)	100
Zelená (00)	1		Oranž (01)	100
Oranž (01)	X		Červená (10)	010
Červená (10)	X		Oranž2 (11)	001
Oranž2 (11)	X		Zelená (00)	010

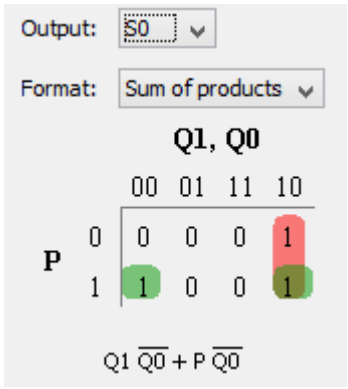
pozn – pozor na výber (kódu) počátečního stavu (stav po resetu, náběhu napájení)

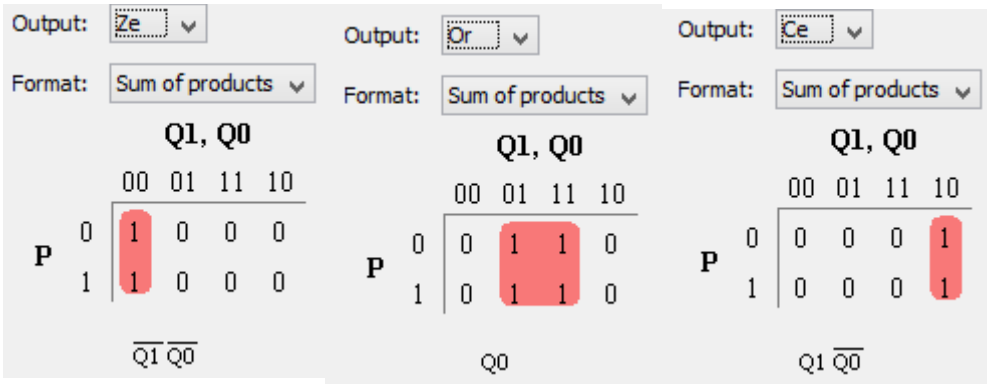
Návrh s D-klopnými obvody

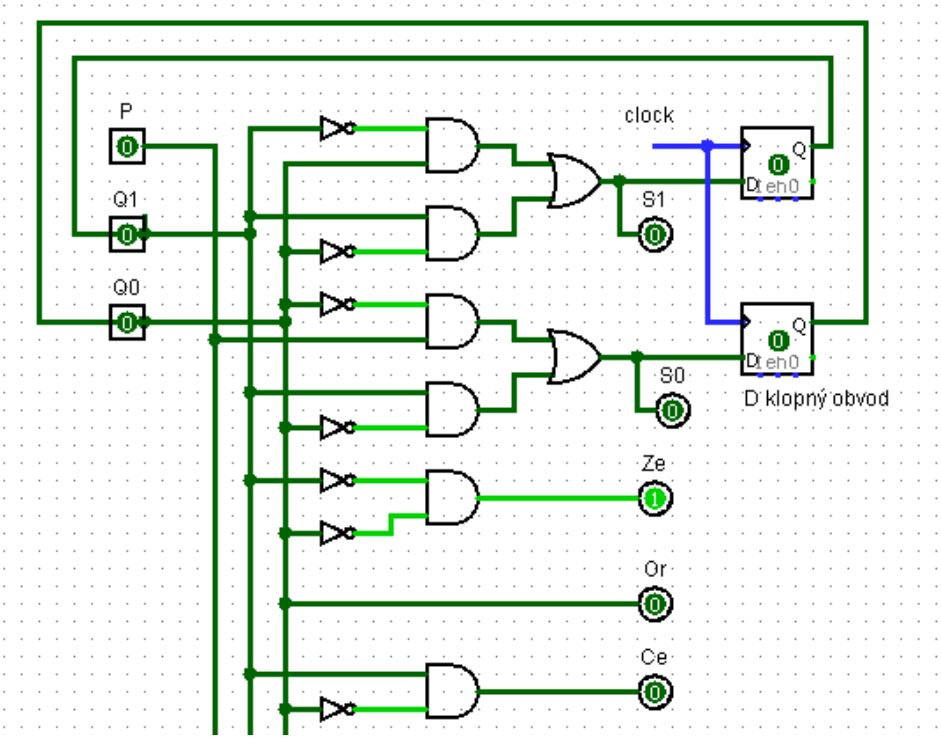
Q1,Q0 = aktuální stav, S1,S0 = nový stav – návrh kombinačního obvodu pro buzení klopných obvodů typu D

Q1	Q0	P	S1	S0	Ze	Or	Ce
0	0	0	0	0	1	0	0
0	0	1	0	1	1	0	0
0	1	0	1	0	0	1	0
0	1	1	1	0	0	1	0
1	0	0	1	1	0	0	1
1	0	1	1	1	0	0	1
1	1	0	0	0	0	1	0
1	1	1	0	0	0	1	0







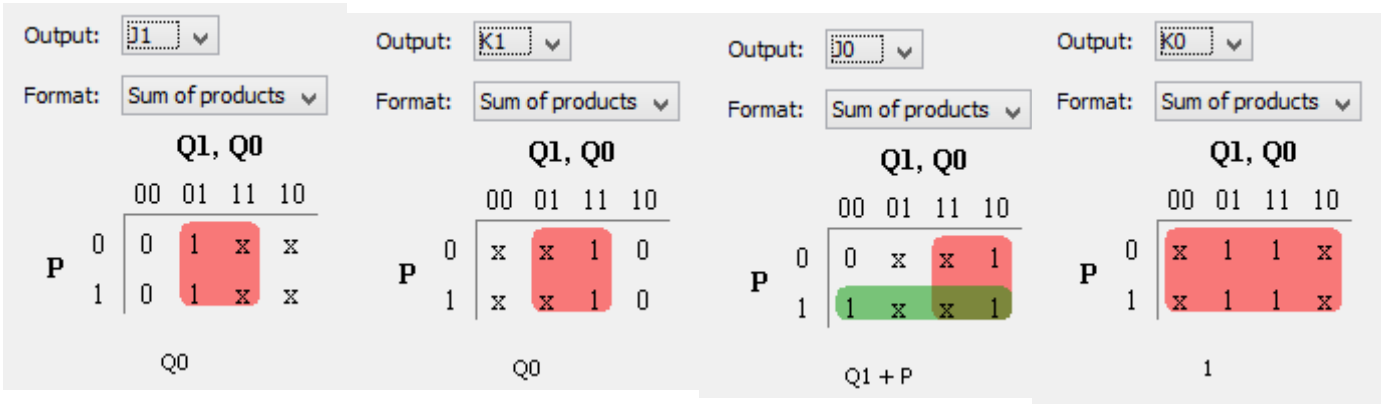


pozn: Kombinační obvod dpolněný o klopné obvody D – dále lze ještě využít výstupu ~Q a odstranit invertory

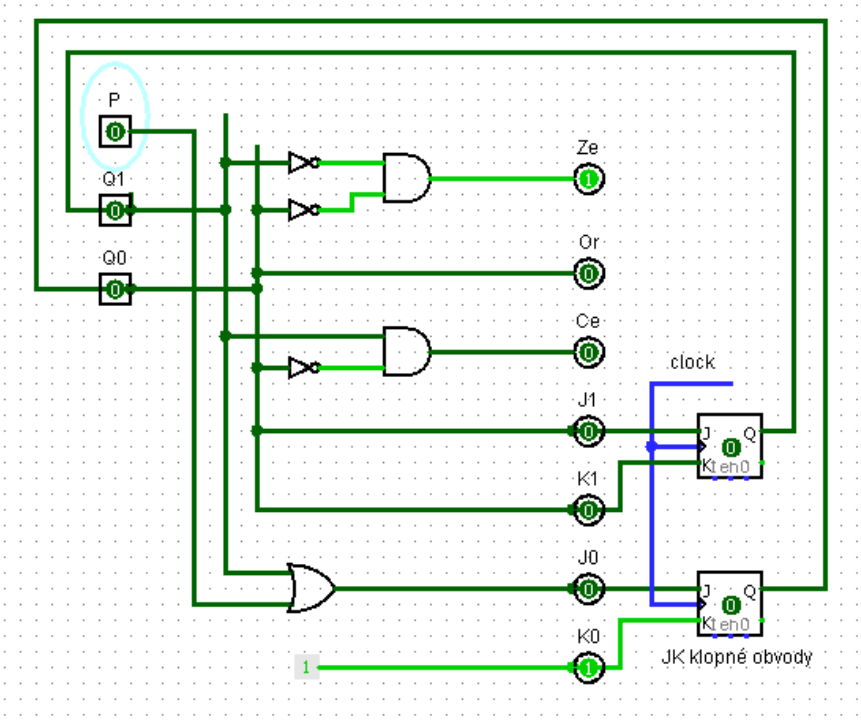
Návrh s JK-klopnými obvody

Vycházíme z požadovaných přechodů Q1→S1 (Q0→S0) a k tomu píšeme příslušné buzení JK klopného obvodu J1K1 (J0K0)

P	Q1	Q0	S1	S0	Ze	Or	Ce	J1	K1	J0	K0
0	0	0	0	0	1	0	0	0	x	0	x
0	0	1	1	0	0	1	0	1	x	x	1
0	1	0	1	1	0	0	1	x	0	1	x
0	1	1	0	0	0	1	0	x	1	x	1
1	0	0	0	1	1	0	0	0	x	1	x
1	0	1	1	0	0	1	0	1	x	x	1
1	1	0	1	1	0	0	1	x	0	1	x
1	1	1	0	0	0	1	0	x	1	x	1



pozn: fce Ze,Or,Ce zůstavají stejné jako u předchozího návrhu, fce S1,S0 uvedeny jen pro srozumitelnost



Kombinační (Sekvenční*) Logické obvody - Stavební prvky (hradla)

- vstup / výstup (0/1, třístavový výstup (Z = vys.impedance), OC (otevřený kolektor))
- napěťová kompatibilita, spotřeba (klidová vs změna, výstupní budič vs vstup)

Standardní logika:

Hradla - Negace, OR. AND, XOR, NAND, NOR

Buffery, Drivery

Registry* - D,JK,Shift,FIFO,Paměti

Spínače, multiplexery:

Převodníky úrovní

Budiče

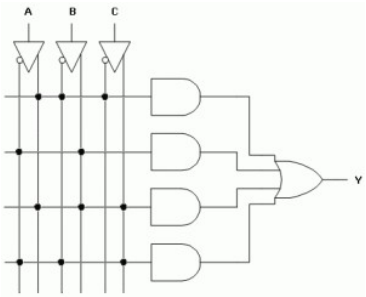
Dekoder (1:N, adresový), Dekodér (BCD, 7segment)

Multiplexer, demultiplexer

Čítače*

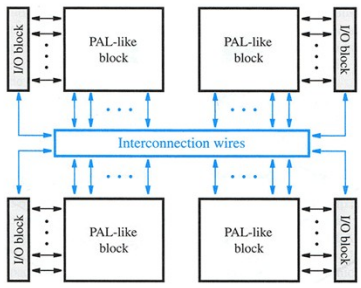
Obvody typu PAL (programmable array logic):

Obvody typu PAL(vnitřní struktura – NEG-AND-OR)

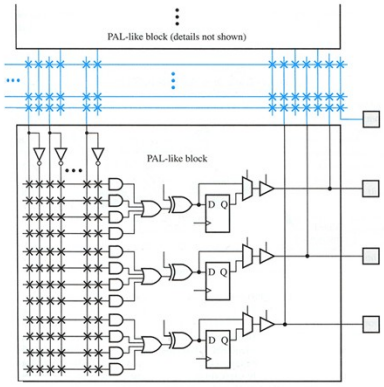


Obvody typu CPLD:

(+dvojitá zpětná vazba, skryté makrobuňky, konfugurovatelné KO, řízení 3-stavových výstupů, spec.podpůrné struktury napr. pro aritmetické operace)



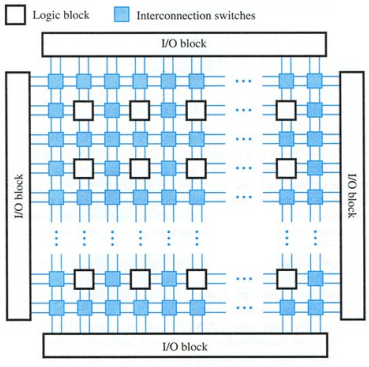
Structure of a complex programmable logic device (CPLD)



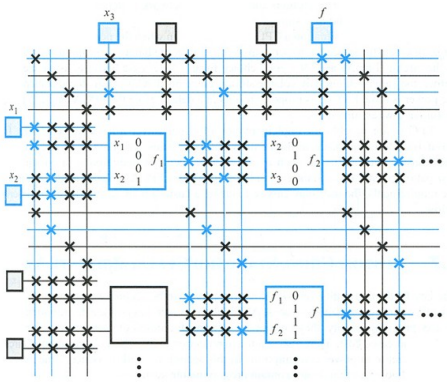
A section of CPLD

Obvody typu FPGA

Pole konfigurovatelných log.bloků (CLB) + programovatelné propojovací struktura (PI) + vstupně výstupní bloky (IOB) + specializované bloky (paměti, násobičky, procesory,...)



General structure of an FPGA



A section of a programmed FPGA