Návrh číslicových systémů (INC)

Otto Fučík

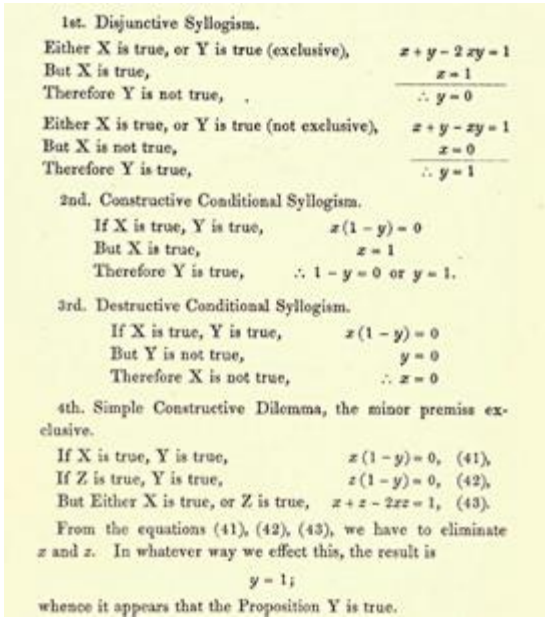
Vysoké učení technické v Brně
Fakulta informačních technologií
Božetěchova 2, 612 66 Brno

Administrativa

- Kontrola znalostí
  - Půlsemestrální zkouška (25 bodů)
  - Projekt (20 bodů) – zápočet
  - Semestrální zkouška (55 bodů)
  - Min. 5 bodů z projektu - podmínka nutná pro získání zápočtu
- Pravidla
  - Pro získání bodů ze semestrální zkoušky je nutné ji vypracovat tak, aby byla hodnocena nejméně 25 body (z celkem 55 bodů); v opačném případě bude přiděleno 0 bodů
  - Pokud bude odhaleno plagiátorství nebo nedovolená spolupráce na projektech, příslušné body nebudou uděleny a bude zváženo zahájení disciplinárního řízení

Návrh číslicových systémů: Historie

- 1822 (Ch. Babbage)
  - Informace mohou být reprezentovány čísly
- 1854 (G. Boole)
  - Matematický aparát umožňující efektivní práci s dvoustavovými (binárními, 0 a 1) funkcemi, výrazy a jejich algebrou
  - Umožňuje systematický návrh a optimalizaci základních stavebních prvků číslicových systémů – tzv. logických obvodů
- 1904 (E. V. Huntington)
  - Rozvinutí a doplnění Booleovy algebry
- 1938 (C. E. Shannon)
  - Využití Booleovy algebry pro návrh log. obvodů (diplomová práce)
  - Použití relé pro realizaci logických operací - sepnuto a rozepnuto (0 a 1)



1st. Disjunctive Syllogism.

Either X is true, or Y is true (exclusive), $x + y - 2xy = 1$
But X is true, $x = 1$
Therefore Y is not true, $\therefore y = 0$

Either X is true, or Y is true (not exclusive), $x + y - xy = 1$
But X is not true, $x = 0$
Therefore Y is true, $\therefore y = 1$

2nd. Constructive Conditional Syllogism.

If X is true, Y is true, $x(1 - y) = 0$
But X is true, $x = 1$
Therefore Y is true, $\therefore 1 - y = 0$ or $y = 1$.

3rd. Destructive Conditional Syllogism.

If X is true, Y is true, $x(1 - y) = 0$
But Y is not true, $y = 0$
Therefore X is not true, $\therefore x = 0$

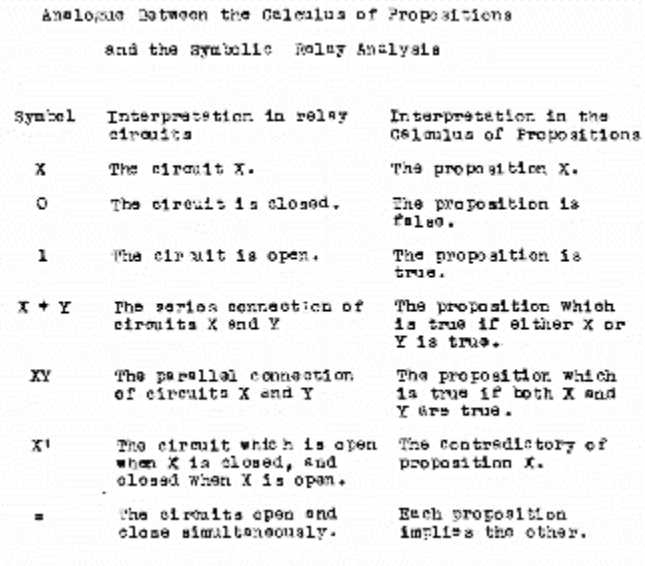
4th. Simple Constructive Dilemma, the minor premiss exclusive.

If X is true, Y is true, $x(1 - y) = 0$, (41),
If Z is true, Y is true, $z(1 - y) = 0$, (42),
But Either X is true, or Z is true, $x + z - 2xz = 1$, (43).

From the equations (41), (42), (43), we have to eliminate x and z. In whatever way we effect this, the result is

$$y = 1;$$

whence it appears that the Proposition Y is true.

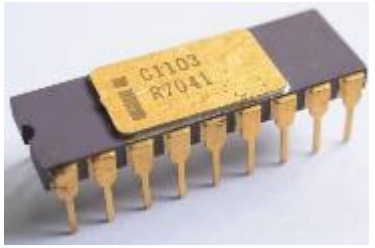


Analogue Between the Calculus of Propositions and the Symbolic Relay Analysis

Symbol	Interpretation in relay circuits	Interpretation in the Calculus of Propositions
X	The circuit X.	The proposition X.
0	The circuit is closed.	The proposition is false.
1	The circuit is open.	The proposition is true.
X + Y	The series connection of circuits X and Y	The proposition which is true if either X or Y is true.
XY	The parallel connection of circuits X and Y	The proposition which is true if both X and Y are true.
X'	The circuit which is open when X is closed, and closed when X is open.	The contradictory of proposition X.
=	The circuits open and close simultaneously.	Each proposition implies the other.

Návrh číslicových systémů: Rozvoj elektroniky

- Efektivní tvorba (rozměry, spolehlivost, příkon) elektronických obvodů pracujících s binárními hodnotami
- 1906 (L. de Forest)
  - Vynález elektronky (zesilování signálů)
- 1947 (W. B. Shockley, J. Bardeen a W. H. Brattain)
  - Vytvoření tranzistoru (rozměry, příkon, cena, spolehlivost)
- 1958 (J. Kilby)
  - Vynález integrovaných obvodů (IO) - umožnil umístění mnoha tranzistorů (dnes miliardy) na polovodičovou destičku
- 1966 (R. Dennard)
  - Vynález pamětí DRAM - realizace spolehlivých a rychlých elektronických pamětí s velkou kapacitou
- Budoucnost? (nanotechnologie, kvantové jevy…)



Mooreův zákon: Predikce z roku 1965

- Exponenciální růst složitosti čipů
- „Počet tranzistorů, které mohou být umístěny na integrovaný obvod, se při zachování stejné ceny zdvojnásobí zhruba každých 18 měsíců“
- Dnešní technologie výroby umožňují integrovat miliardy tranzistorů v jednom čipu
  - Umíme je efektivně využít?

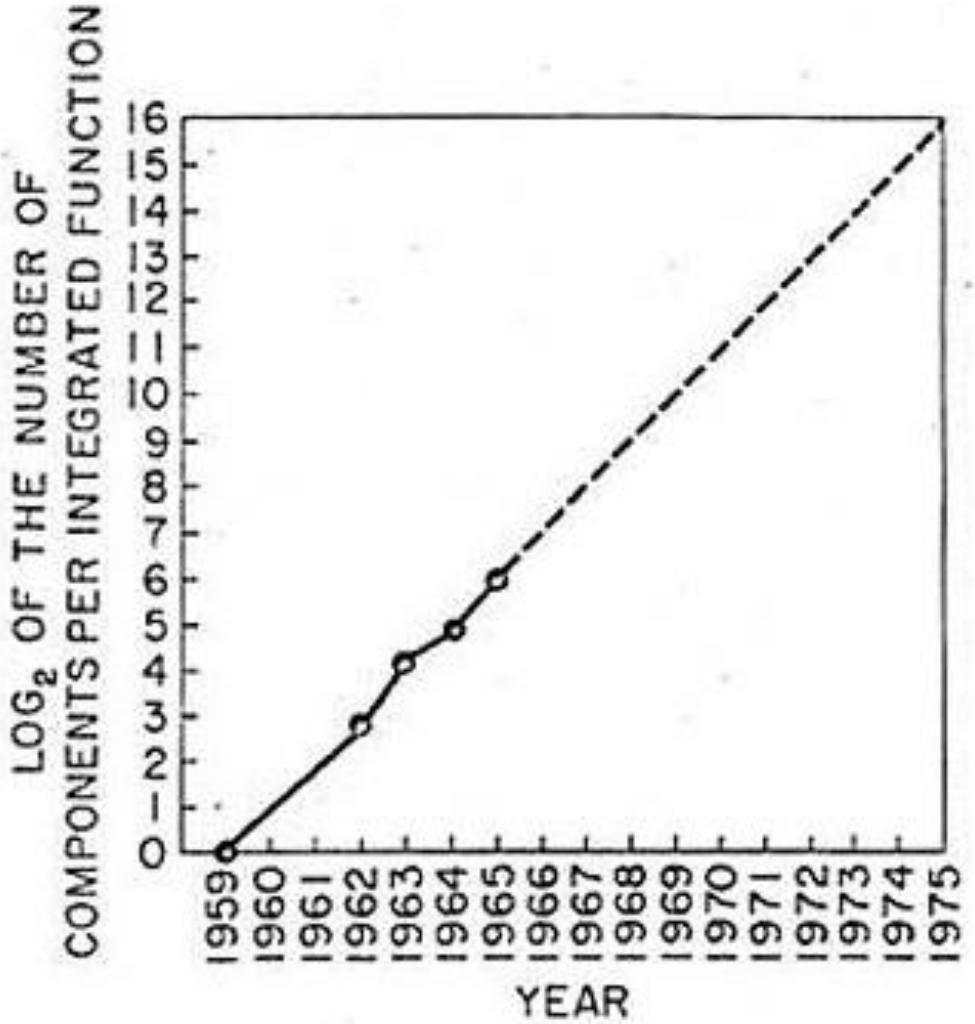
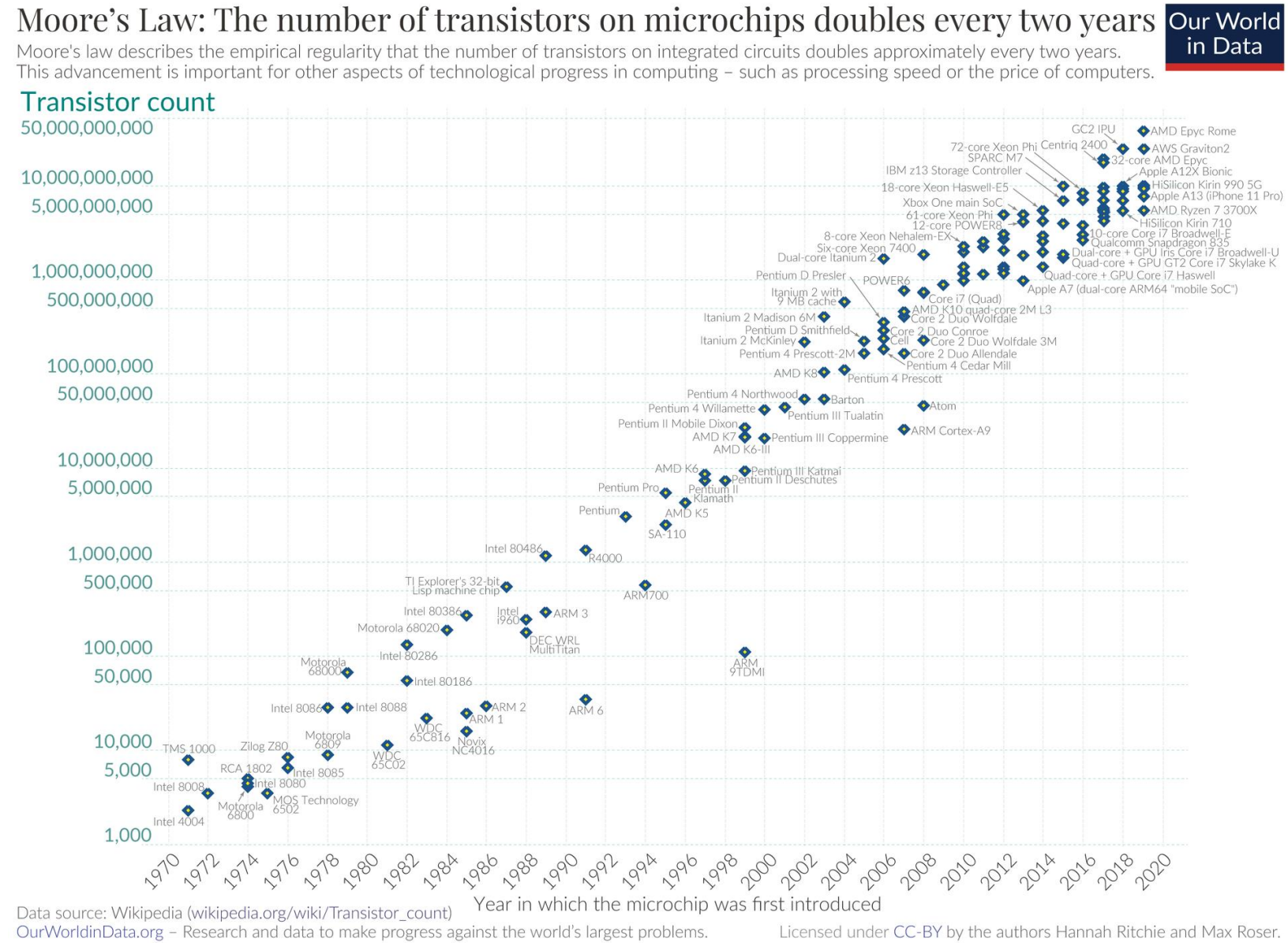


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

[Zdroj: http://www.intel.com/pressroom/kits/events/moores_law_40th/index.htm?iid=tech_mooreslaw+body_presskit]

Mooreův zákon: Stav v roce 2022



Semiconductor device fabrication

MOSFET scaling (process nodes)

- 10 µm – 1971
- 6 µm – 1974
- 3 µm – 1977
- 1.5 µm – 1981
- 1 µm – 1984
- 800 nm – 1987
- 600 nm – 1990
- 350 nm – 1993
- 250 nm – 1996
- 180 nm – 1999
- 130 nm – 2001
- 90 nm – 2003
- 65 nm – 2005
- 45 nm – 2007
- 32 nm – 2009
- 22 nm – 2012
- 14 nm – 2014
- 10 nm – 2016
- 7 nm – 2018
- 5 nm – 2020
- 3 nm – 2022

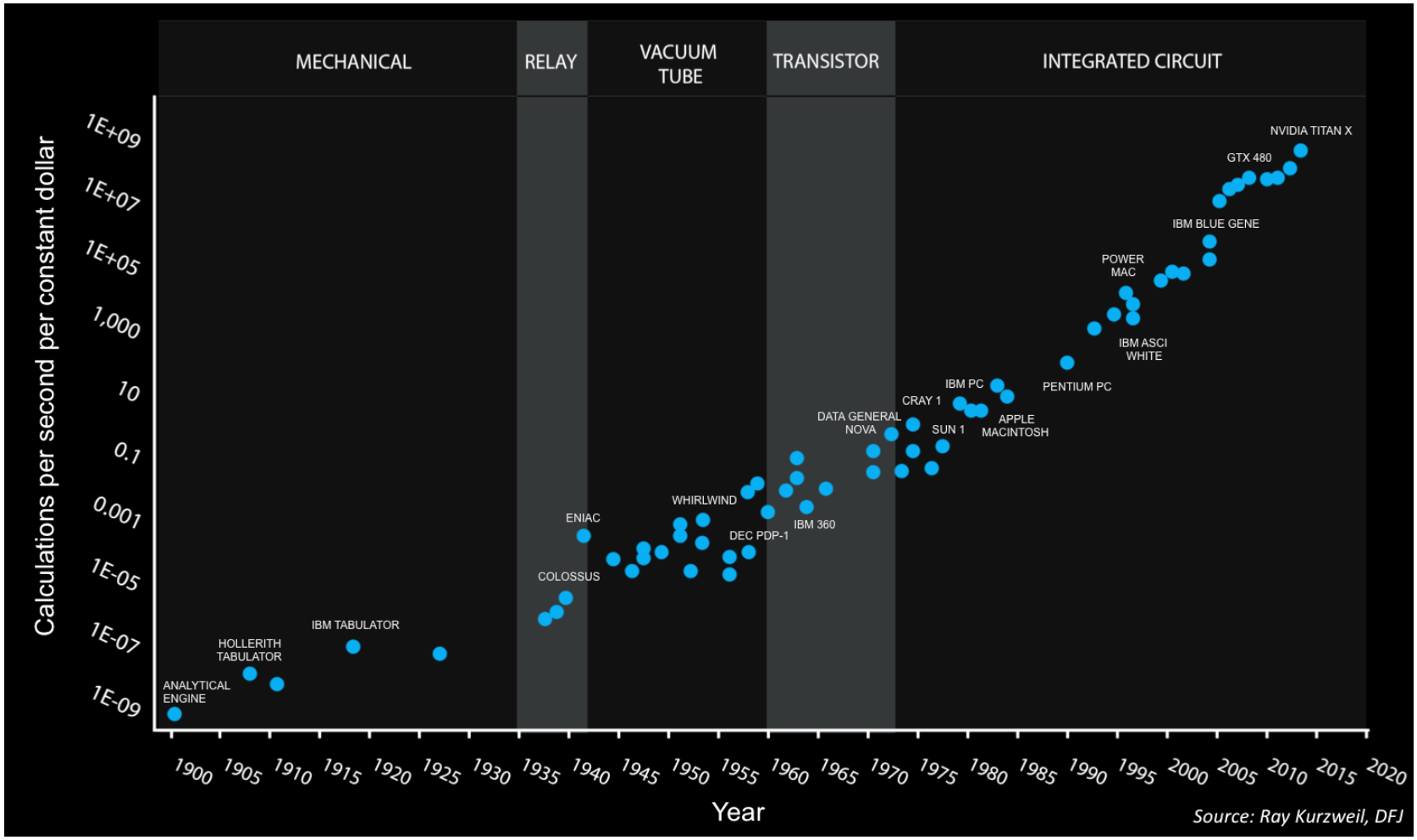
Future

- 2 nm ~ 2024

Zdroje: https://en.wikipedia.org/wiki/Moore%27s_law https://en.wikipedia.org/wiki/Transistor_count

Mooreův zákon: Extrapolace do minulosti

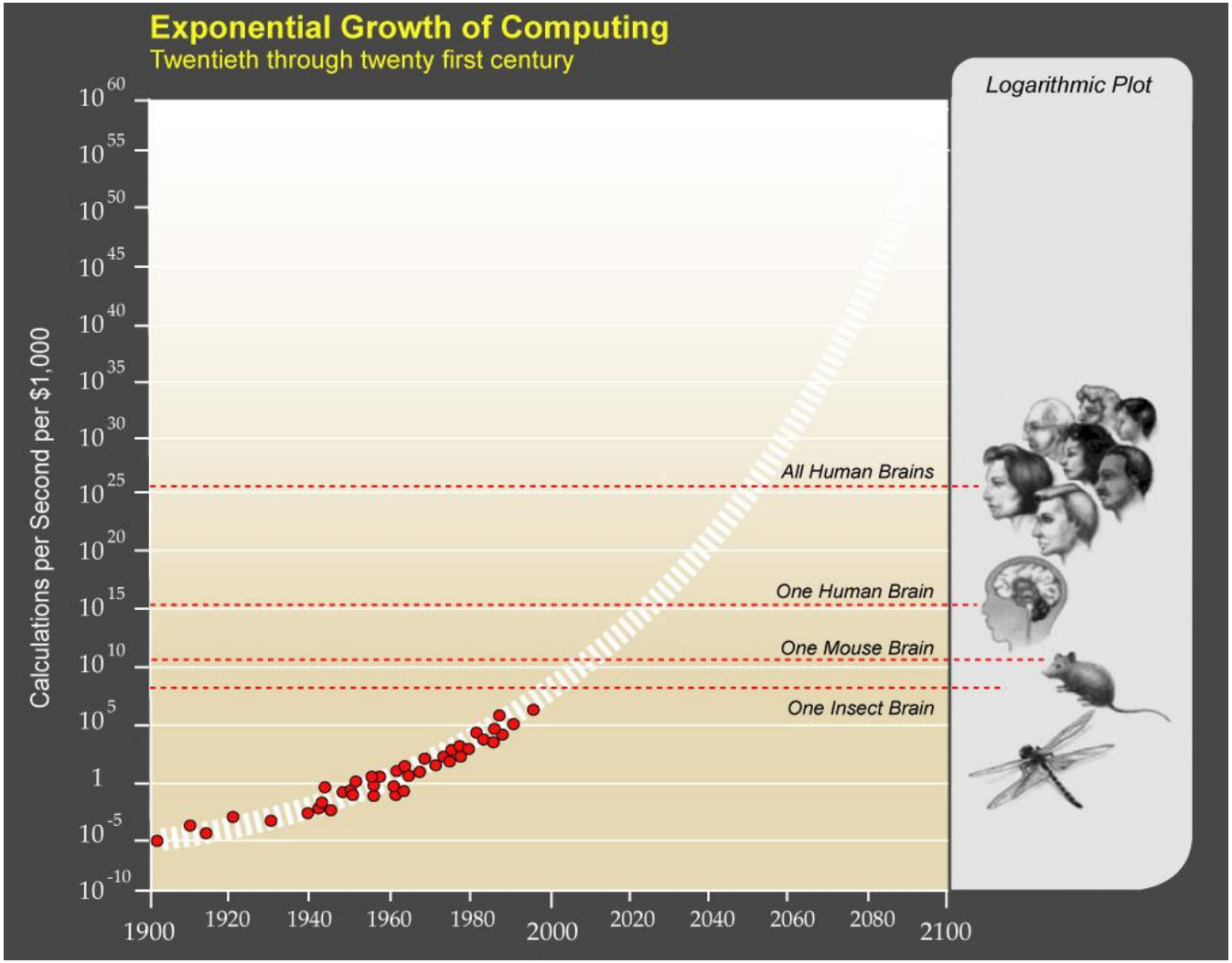
- Srovnání s výpočetními stroji, které byly sestrojeny před vynálezem tranzistoru



[Zdroj: https://en.wikipedia.org/wiki/Accelerating_change#/media/File:Moore's_Law_over_120_Years.png]

Mooreův zákon: Extrapolace do budoucnosti

- Spekulativní srovnání s „biologickými výpočetními stroji“



[Zdroj: https://en.wikipedia.org/wiki/Accelerating_change#/media/File:PPTExponentialGrowthof_Computing.jpg]

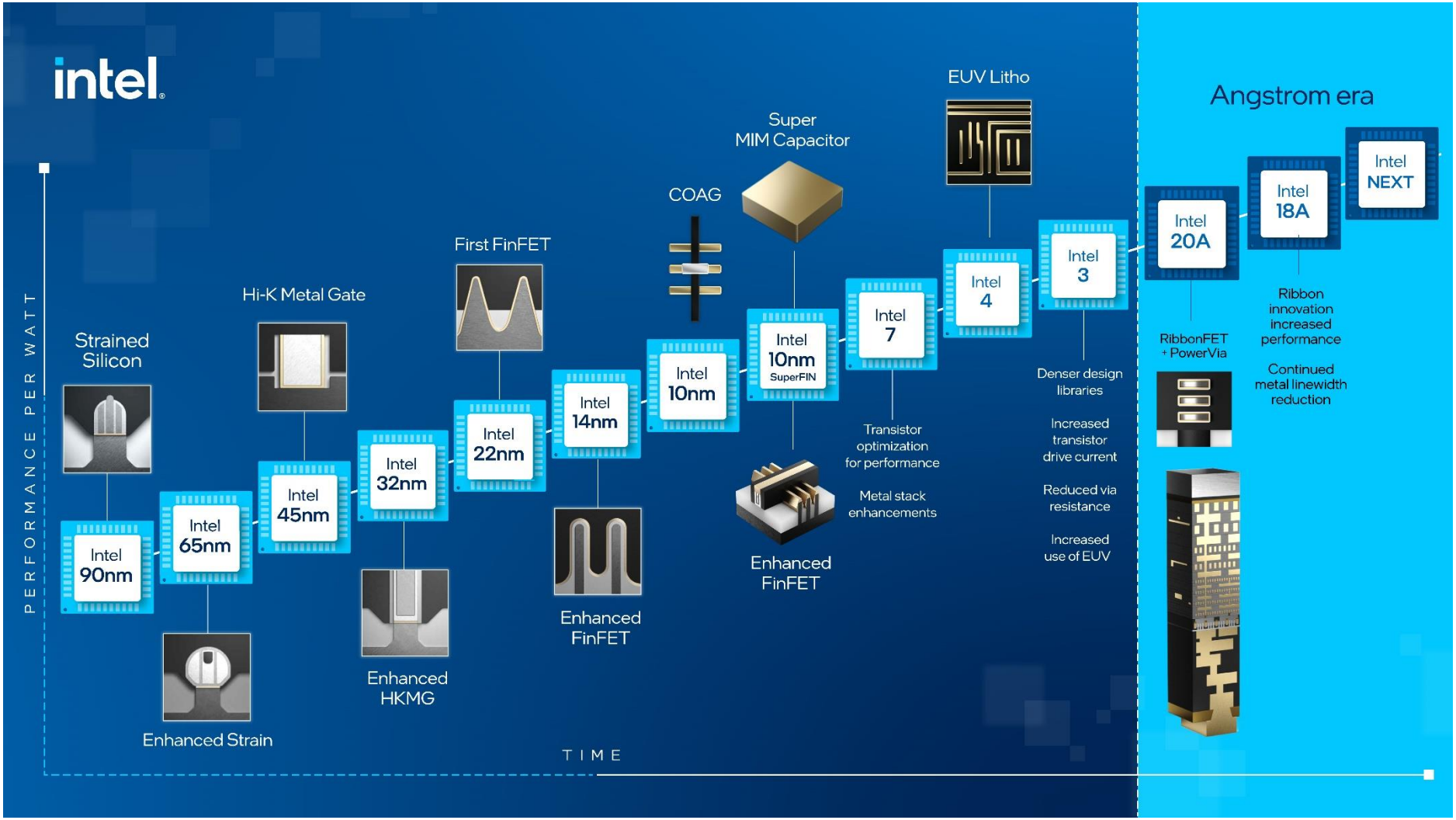
Mooreův zákon: AI revoluce

- Čip pro AI Cerebras WSE-2 má 2,6 bilionů (10^{12}) tranzistorů
  - Největší GPU má „pouze“ 54,2 miliard (10^{9}) tranzistorů
  - Využívá maximální plochu křemíku, kterou lze vyrobit z Si plátku o průměru 300 mm
  - Obsahuje 850.000 AI jader a paměť 40 GB

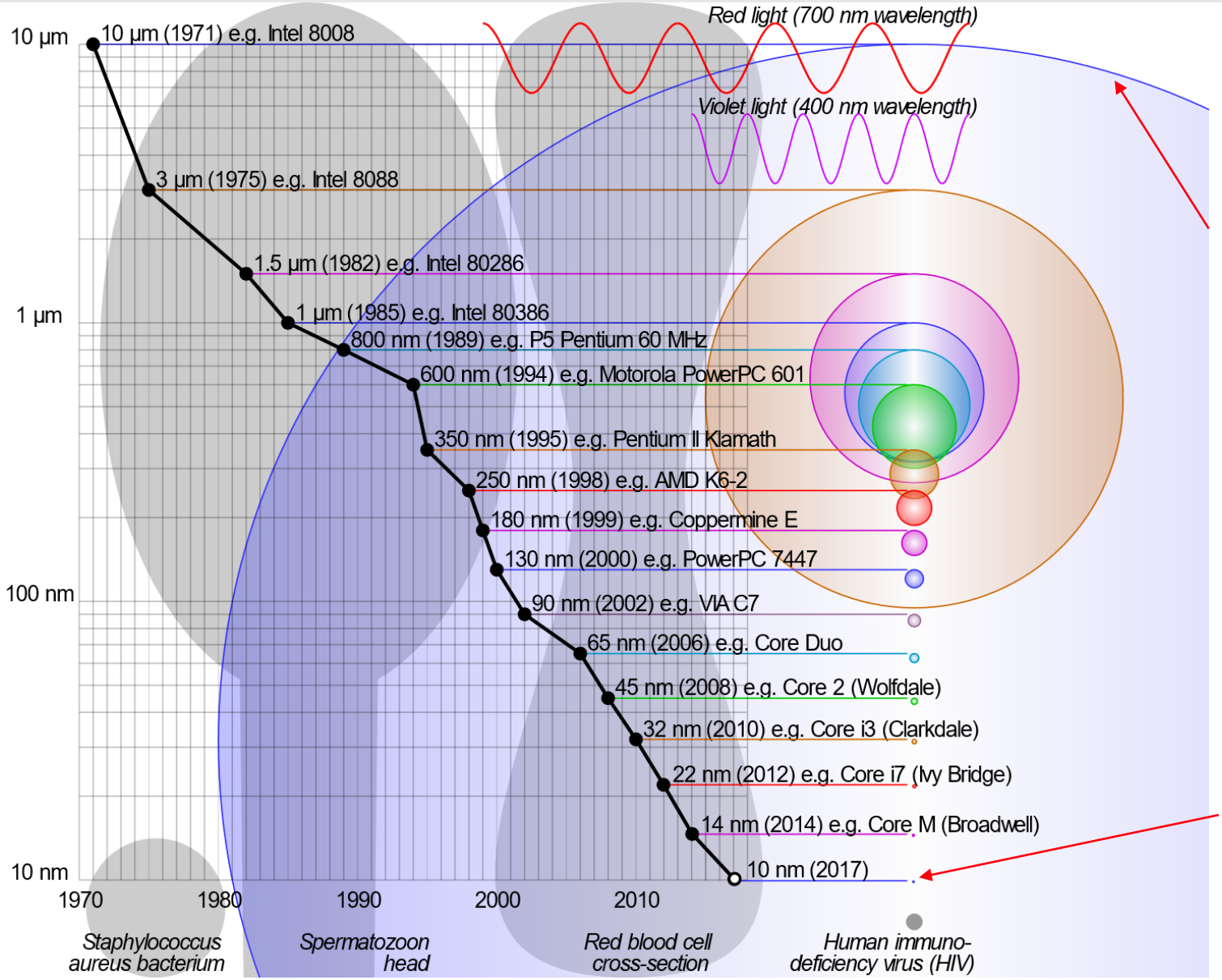
[Zdroj: https://www.cerebras.net/product-chip/]

Mooreův zákon: Budoucnost dle firmy INTEL

- 1 Angstrom = 100 pm = 0,1 nm



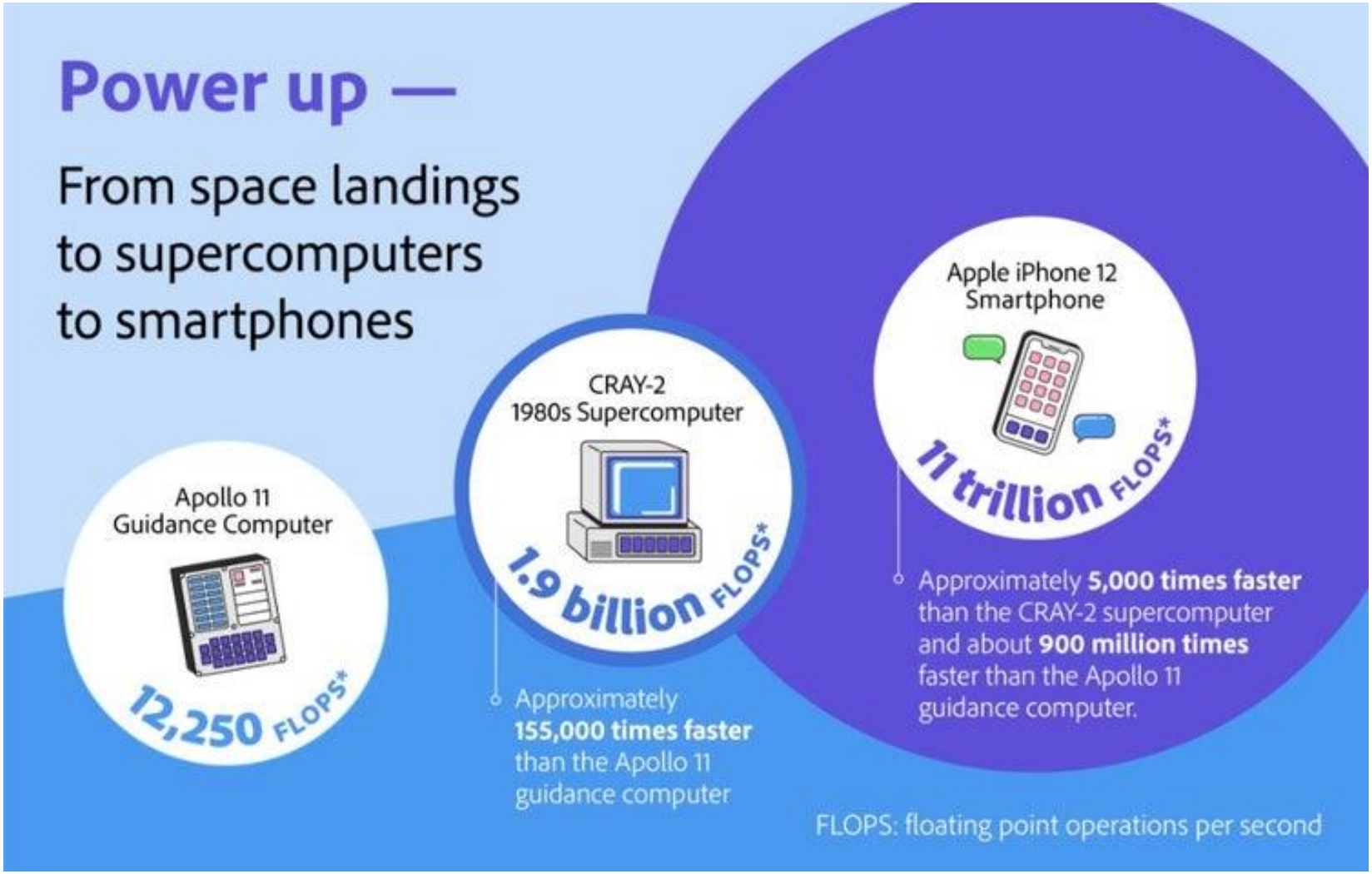
Mooreův zákon: Ilustrace zmenšování rozměrů



[Zdroj:By Cmglee - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=16991155]

Mooreův zákon: Ilustrace důsledků

- Růst výpočetní výkonnosti



[Zdroj: https://blog.adobe.com/en/publish/2022/11/08/fast-forward-comparing-1980s-supercomputer-to-modern-smartphone]

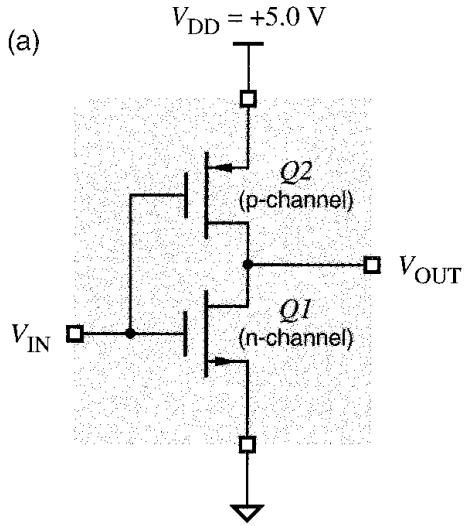
Mooreův zákon: Ilustrace důsledků

- Zmenšují se rozměry a klesá hmotnost

[Zdroj: https://blog.adobe.com/en/publish/2022/11/08/fast-forward-comparing-1980s-supercomputer-to-modern-smartphone]

Logický člen technologie CMOS Invertor

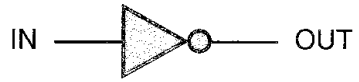
- Logický člen „invertor“
  - V technologii CMOS je sestaven ze dvou komplementárních tranzistorů MOSFET
  - Na základě velikosti vstupního napětí je tranzistor Q1 buď sepnut a Q2 rozepnut, nebo naopak
- Pro ilustraci si můžeme jejich funkci modelovat pomocí ideálních spínačů

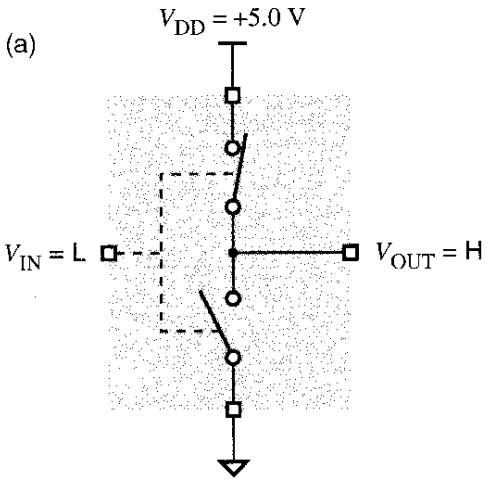


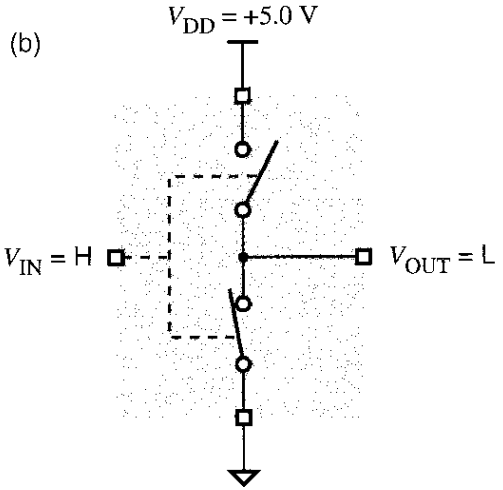
(b)

V_{IN}	Q1	Q2	V_{OUT}
0.0 (L)	off	on	5.0 (H)
5.0 (H)	on	off	0.0 (L)

(c)

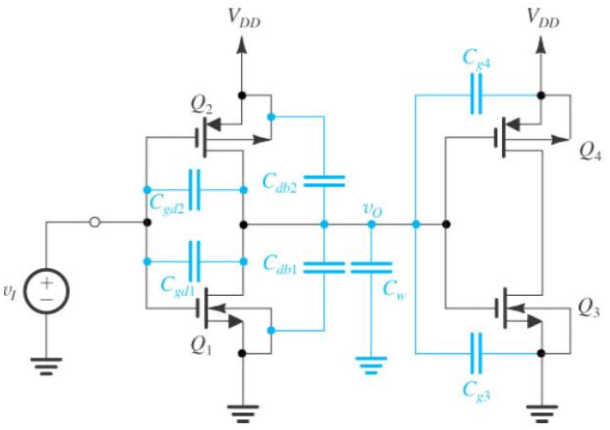


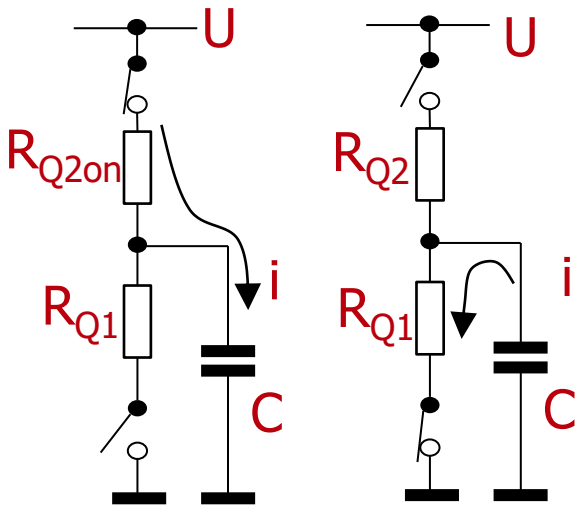




Logické členy technologie CMOS: Invertor

- Tranzistory nejsou ideální spínače
  - Nemají při sepnutí nulový vnitřní odpor R_Q a při rozepnutí nekonečně velký izolační odpor
- Tranzistory a spoje mezi logickými členy mají nenulové parazitní kapacity C
- Parazitní kapacity je třeba nabíjet a vybíjet proudem i
- Průchodem proudu i přes odpory R_Q vzniká na těchto odporech úbytek napětí $U=i*R_Q$
- Příkon obvodu: $P=U*i= i^2*R_Q$
- Dodaná energie se promění v teplo, které je třeba z obvodu odvést

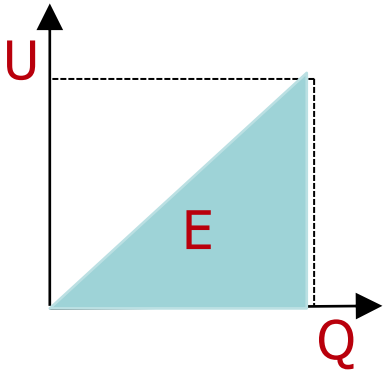


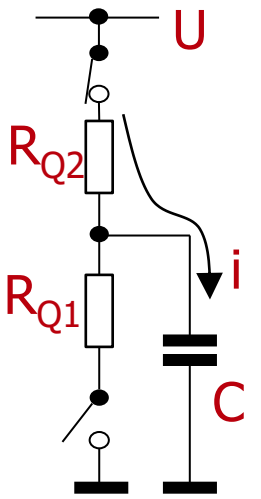


Limity: Příkon a energie

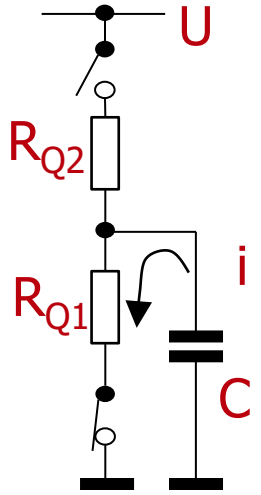
- Náboj Q kondenzátoru C: $Q = C \cdot U\ [C] = I \cdot t\ [As]$
- Energie potřebná pro nabití kondenzátoru C:

$$E = 1/2 \cdot Q \cdot U = 1/2 \cdot C \cdot U^2\ [J = Ws]$$





- Nabití C: $E_C = E_{RQ1} = 1/2 \cdot C \cdot U^2$
- Vybití C: $E_C = E_{RQ2} = 1/2 \cdot C \cdot U^2$
- Cyklus nabití a vybití: $E_{0\to1\to0} = E_{RQ1} + E_{RQ2} = C \cdot U^2$
- Dynamický příkon potřebný pro periodické nabíjení a vybíjení C: $P[W] = \frac{E[Ws]}{t[s]} = E[Ws] \cdot f[Hz] = C \cdot U^2 \cdot f$
- f…frekvence změn 0-1-0 (frekvence hodinového signálu)

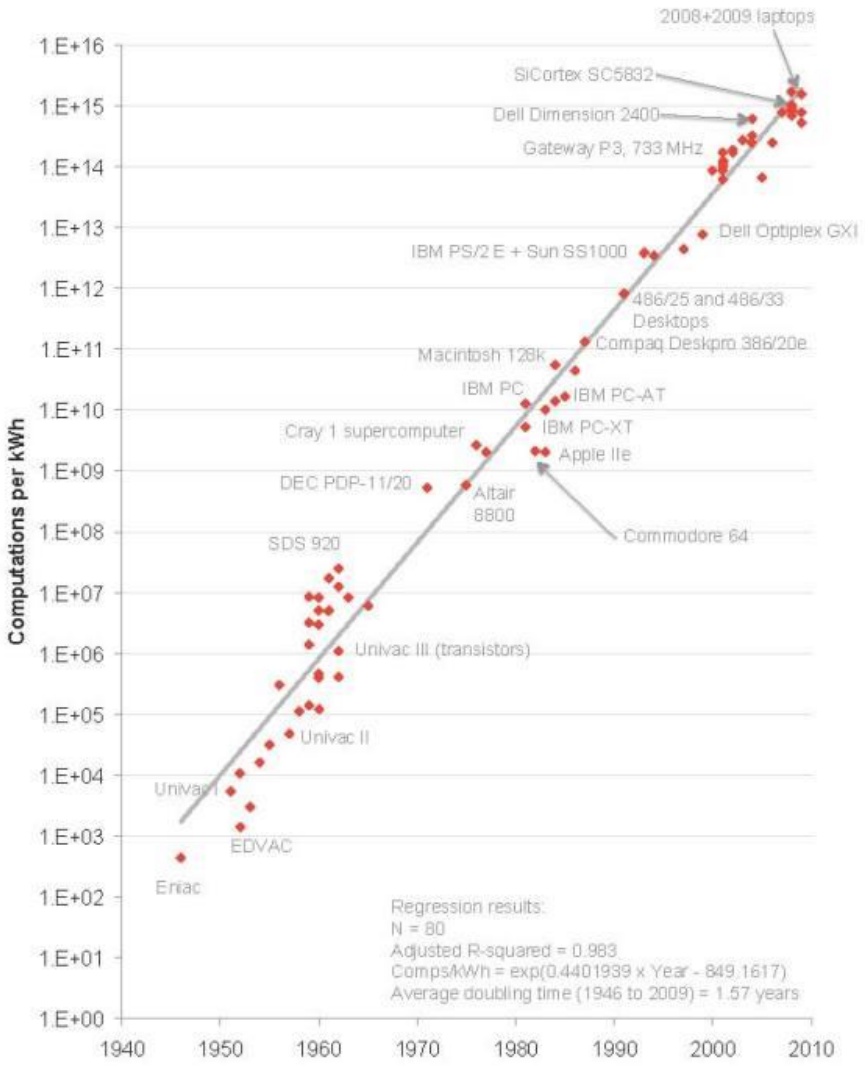


Limity: Výpočetní výkon a energetická náročnost

- Pollackovo pravidlo
  - Skutečně využitelný výpočetní výkon je zhruba úměrný druhé odmocnině počtu tranzistorů - neroste tedy lineárně se zvyšováním počtu tranzistorů (je to dáno omezeními v architektuře výpočetních obvodů)
  - 4x více tranzistorů ~ cca 2x větší výpočetní výkon
  - Spotřeba energie však (při konstantním rozměru tranzistoru) roste zhruba lineárně s nárůstem počtu tranzistorů
  - 4x více tranzistorů ~ cca 4x větší spotřeba energie
- Mooreův zákon
  - Počet tranzistorů roste cca 2x za 1,5 roku
  - Rozměr tranzistoru je cca poloviční - plocha tranzistorů, kterou tranzistor zabírá tedy klesá cca 4x za 1,5 roku
  - Menší plocha tranzistoru vede na menší parazitní kapacity, což vede snižuje dynamický příkon tranzistorů

Limity: Výpočetní výkon a energetická náročnost

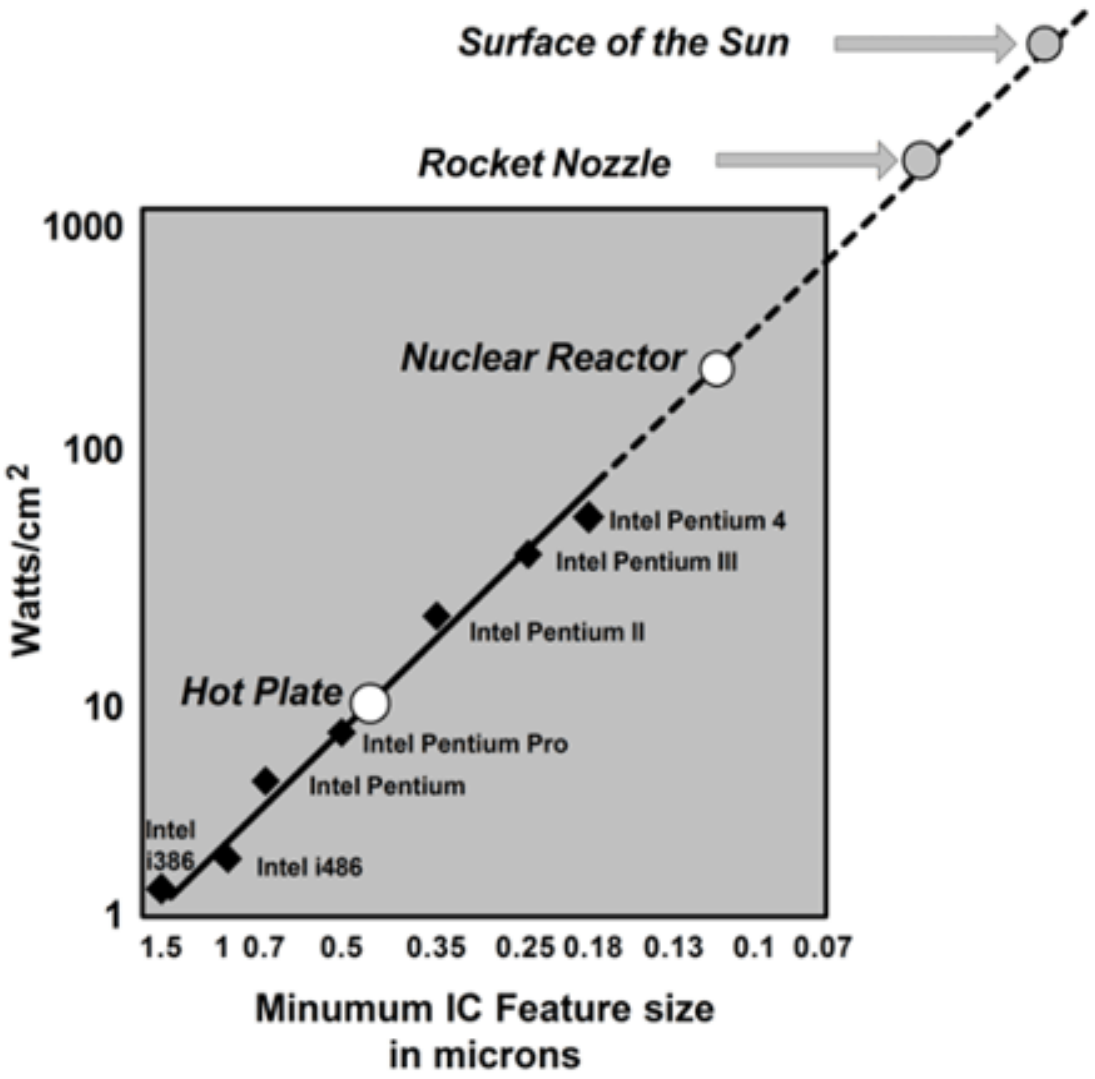
- Koomeyův zákon
  - Množství výpočtů na jednotku energie se zdvojnásobí přibližně každého 1,6 roku



[Zdroj: https://en.wikipedia.org/wiki/Koomey%27s_law]

Limity: Odvod tepla

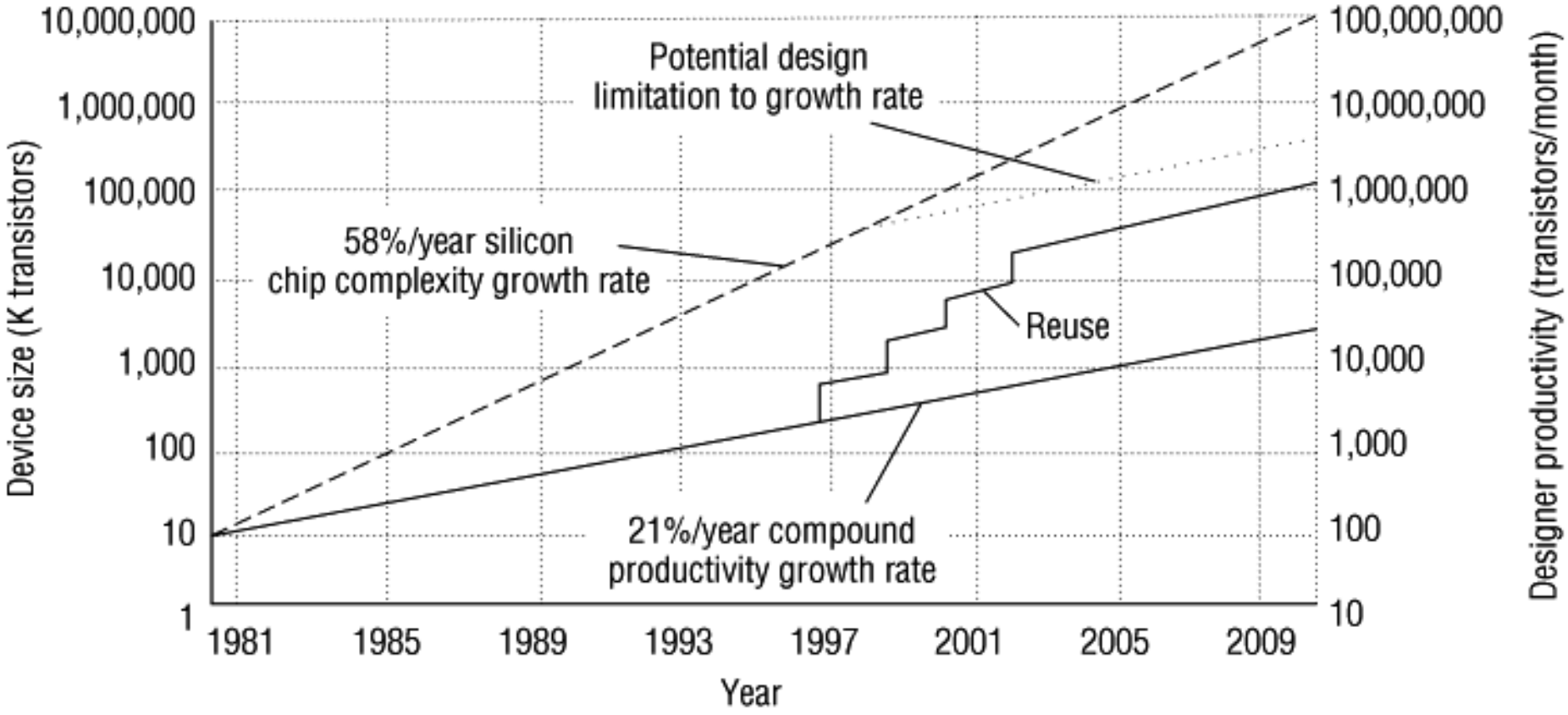
- Hustota energie [W/cm^2] roste s přibližně lineárně s výkonností
- Dodaná energie se přemění v teplo, které se musí odvést
- Odvod tepla (chlazení) je technologicky náročné či dokonce nemožné, což limituje dosažitelnou výpočetní výkonnost



[Zdroj: R. Ronen, A. Mendelson, K. Lai, Shih-Lien Lu, F. Pollack and J. P. Shen, "Coming challenges in microarchitecture and architecture," in Proceedings of the IEEE, vol. 89, no. 3, pp. 325-340, March 2001]

Limity: Produktivita práce při návrhu

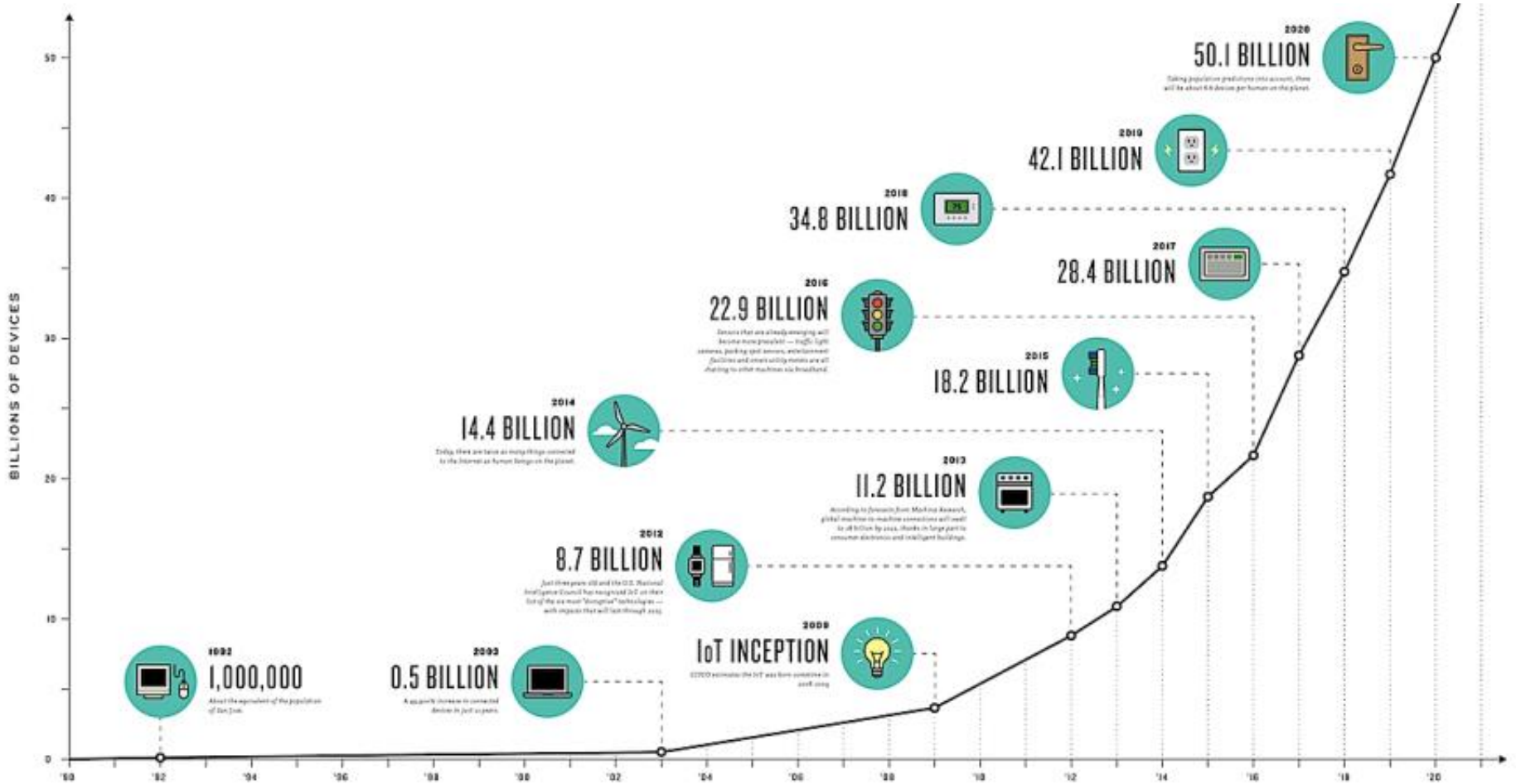
- Složitost integrovaných obvodů roste o cca 60 % ročně
- Produktivita práce pouze o cca 20 % ročně



[Zdroj: B. Smith, "Burton Smith's Multithreaded Success Strategy" in IEEE Design & Test of Computers, vol. 16, no. 04, pp. 7-13, 1997]

Roste jak složitost komponent, tak jejich počet

- Internet věcí (Internet of Things – IoT)
  - Propojené vestavěné systémy pro sběr a výměnu dat



[Zdroj: J. Straw: http://idisrupted.com/disrupted-electronics-internet-things-may-create-moores-law-steroids/]

Číslicové vs. analogové systémy

- Analogové systémy nejsou pro řešení některých úloh použitelné
  - Např. nelineární funkce (šifrování apod.)
- Číslicové systémy lze programovat pro vykonávání libovolného vyčíslitelného algoritmu (omezeno velikostí paměti a dobou výpočtu)
- Informace může být ve fyzicky realizovaném číslicovém systému reprezentována s mnohem větší přesností a ve větším rozsahu hodnot než v analogovém (signál/šum)
  - Přesnost a rozsah hodnot mohou být teoreticky libovolné (omezeno velikostí paměti a dobou výpočtu)
- Činnost analogových obvodů je v praxi výrazněji limitována řadou fyzikálních veličin
  - Šum, teplotní výkyvy, stárnutí součástek apod.

Číslicové vs. analogové systémy

- Informace reprezentovaná číslicově se lépe ukládá a čte
- Číslicové obvody umožňují realizovat detekci chyb a jejich opravu
- Pro stejnou posloupnost vstupních hodnot produkuje číslicový systém vždy stejné výsledky
  - Např. viz klasická gramofonová deska vs. kompaktní disk
- Návrh číslicových systémů (nazývaných též "logické systémy") pracuje pouze s dvouhodnotovými veličinami a logickými vztahy mezi nimi (Booleova algebra)
  - Číslicové systémy lze navrhovat, analyzovat a realizovat se znalostí relativně jednoduchých principů (náplň tohoto kurzu)
  - Návrh analogových obvodů vyžaduje hlubokou znalost funkce (matematických modelů) použitých součástek

Návrh složitých systémů

- Číslicové systémy
  - Hierarchicky uspořádaná struktura, ve které jednotlivé subsystémy (komponenty) přenáší přes rozhraní informaci pomocí komunikačního média (vodiče)
  - S ohledem na zjednodušení návrhu je výhodné, když komponenty a komunikace mezi nimi je co nejjednodušší => 1bitové hodnoty
  - Proto tvoříme číslicové systémy sestavené z komponent, které pracují a komunikují pouze s 1bitově reprezentovanou informací (nejjednodušší možná forma = jednoduché, spolehlivé, levné…)
  - Umožňují efektivní tvorbu výpočetních strojů (počítače), které zpracovávají binárně kódované informace pomocí logických obvodů (Booleova algebra)

Booleova algebra [G. Boole, 1854]

1. Distributivní komplementární svaz
   - Obsahuje alespoň dva prvky
2. Šestice (B, +, ·, ’, 0, 1)
   - B neprázdná množina s alespoň dvěma různými prvky
   - + logický součet (binární operace)
   - · logický součin (binární operace)
   - ’ komplement (unární operace)
   - 0 nejmenší (nulový) prvek (infimum)
   - 1 největší (jedničkový) prvek (supremum)
   - Definuje množinu prvků, množinu operátorů, axiomy (postuláty) a teorémy (věty)

- Dvouhodnotová Booleova algebra
  - Axiomy a teorémy Booleovy algebry (1854) jsou definovány obecně
  - My se omezíme na algebru, ve které logické proměnné a výsledky logických funkcí mohou nabývat pouze hodnot 0 a 1 (0≠1)

Logické operace

- Základní
  - Logický součet (disjunkce, spojení, sjednocení, OR): $x \vee y,\ x + y$
  - Logický součin (konjunkce, průsek, průnik, AND): $x \wedge y,\ x \cdot y$
  - Negace (inverze, doplněk, komplement, NOT): $x', \overline{x}, \neg x, \sim x$
- Shefferova funkce (negace log. součinu, NAND): $x \uparrow y,\ \overline{x \cdot y}$
- Pierceova funkce (negace log. součtu, NOR): $x \downarrow y,\ \overline{x + y}$
- Exkluzivní log. součet: $x \oplus y,\ \overline{x \equiv y},\ \overline{x} \cdot y + x \cdot \overline{y}$
  - Pro 2 proměnné též nonekvivalence (součet modulo 2)
- Ekvivalence (totožnost, rovnost, XNOR): $x \Leftrightarrow y,\ x \equiv y,\ \overline{x} \cdot \overline{y} + x \cdot y$
- Implikace: $x \Rightarrow y,\ x \rightarrow y,\ \overline{x} + y$
- Inhibice (negace implikace) $\overline{x \Rightarrow y},\ x \nrightarrow y,\ x \cdot \overline{y}$
- Kontradikce: výsledkem je konstanta 0 (nezávisle na vstupech)
- Tautologie: výsledek je konstanta 1 (nezávisle na vstupech)
- Logické operace jsou realizovány logickými členy

Princip duality

- Pokud platí nějaké tvrzení, tak platí i duální tvrzení, které vznikne vzájemnou záměnou operací „+" a „·" a prvků 0 a 1

$$0 \to 1 \quad 1 \to 0 \quad "+" \to "\cdot" \quad "\cdot" \to "+"$$

- Příklad:

$$a + (b \cdot c) = (a + b) \cdot (a + c) \to a \cdot (b + c) = a \cdot b + a \cdot c$$

- Poznámka
  - Pokud platí jisté tvrzení, není třeba dokazovat tvrzení duálního tvrzení

Logické proměnné, funkce a výrazy

- Logické proměnné x_1, x_2,..., x_n
  - Nabývají hodnot 0 a 1 (logické konstanty)
  - Budeme značit písmeny: A, B, C,..., a, b, c, ..., x_1, x_2, x_3...
  - Poznámka: pro jednodušší zápis budeme logický součin značit jak symboly „X·Y", tak „XY" – pozor na záměnu s proměnnou „XY"
- Logická funkce f(x_1, x_2,..., x_n) je zobrazení $f : \{0,1\}^n \rightarrow \{0,1\}$
  - Hodnota log. funkce je buď 1, nebo 0, v závislosti na hodnotách (0, 1) jednotlivých proměnných x_1, x_2,...,x_n
  - Pro *n* proměnných (každá má 2 možné hodnoty) máme 2^n možností, jak jim přiřadit hodnoty
  - Existuje celkem 2^{2^n} různých log. funkcí *n* proměnných
- Logický výraz (řetězec symbolů)
  - Obsahuje log. konstanty, log. proměnné a log. operátory
  - Log. výrazy lze upravovat a zjednodušovat s využitím Booleovy algebry – cílem je splnění daných kritérií (cena, rychlost, příkon atd.)

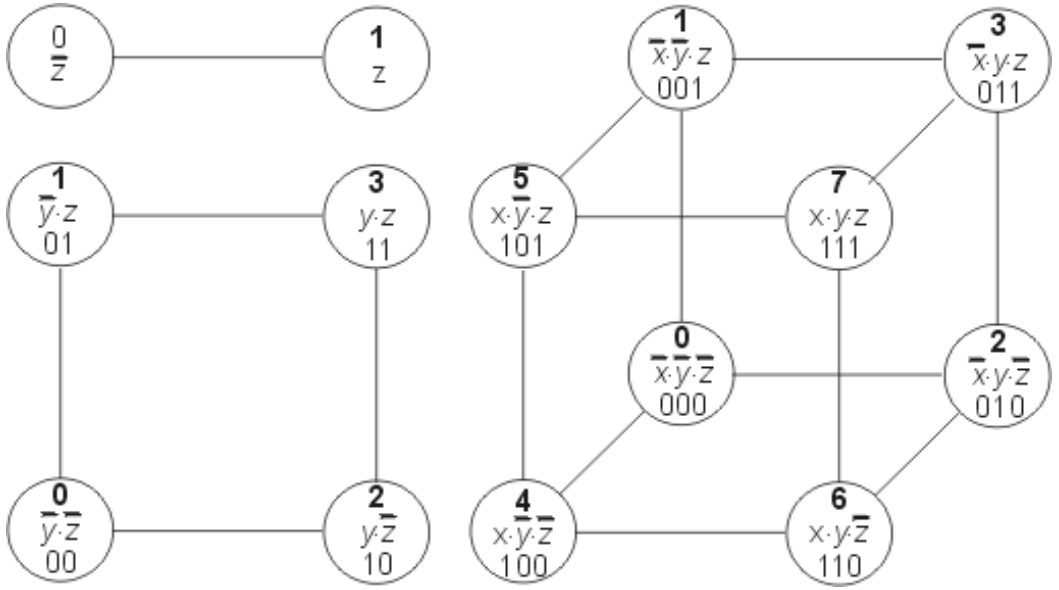
Reprezentace log. funkcí: Pravdivostní tabulka

- Normální báze – (x, y, z)
  - Proměnným jsou přiřazeny váhy mocnin báze 2
- Log. funkce – F(x, y, z)
  - Vstupní stav – kombinace vstupních log. proměnných (vstupů)
  - Stavový index s – desítkové číslo udávající hodnotu log. stavu
  - Neurčený stav X (anglicky don't care)
    - Stav, kdy není třeba určit, zda při daném vstupním stavu má pravdivostní hodnota funkce F(x, y, z) hodnotu 0, nebo 1
  - Log. funkce určená – pro všechny možné vstupní stavy existuje jednoznačné určení pravdivostních hodnot log. funkce F(x, y, z)

stavový index s	vstupní stav			pravdivostní hodnoty
	4	2	1	F(x, y, z)
	x	y	z	
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	X
6	1	1	0	1
7	1	1	1	X

Reprezentace log. funkcí: Jednotková krychle

- Neorientovaný graf
- Reprezentace log. funkce N proměnných
  - S více než čtyřmi rozměry se špatně pracuje
- Uzly
  - Reprezentují hodnoty vstupního stavu
  - Počet uzlů je 2^N
- Hrany
  - Spojují uzly, které se liší pouze v jedné proměnné (sousednost)

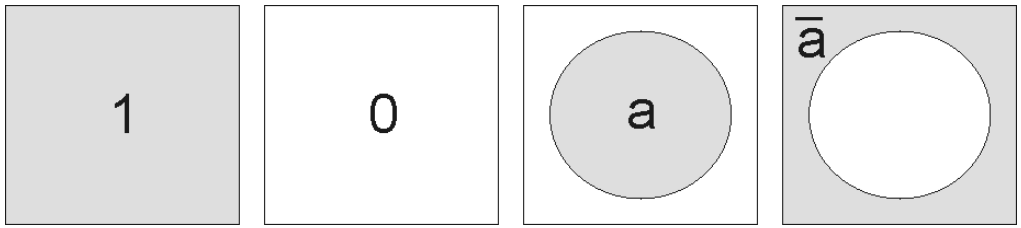


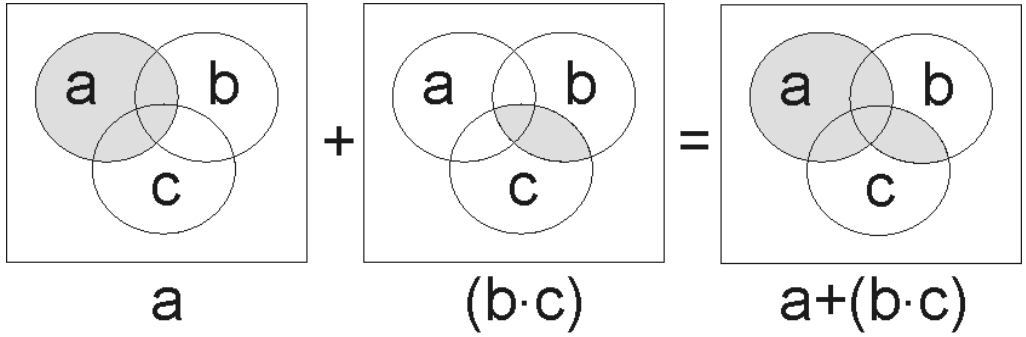
- Použití (bude probráno později)
  - Pro minimalizaci log. funkcí
  - Proměnnou, ve které se liší uzly krychle, lze eliminovat
  - Např.: pro dvourozměrnou krychli a stavy 3 (binárně 11) a 2 (binárně 10) platí:

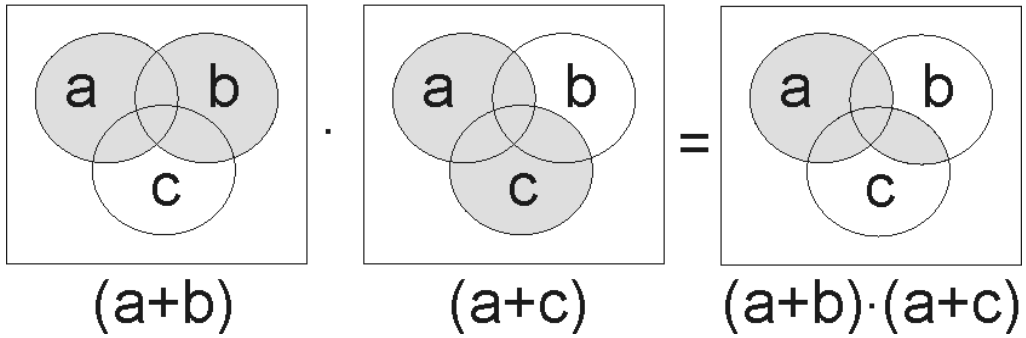
$$yz + y\bar{z} = y$$

Reprezentace log. funkcí: Vennovy diagramy

- Prvky množin jsou znázorněny jako uzavřené plochy
- Log. součet = sjednocení příslušných ploch
- Log. součin = průnik příslušných ploch
- Příklad:
  - Reprezentace distributivního zákona pomocí tzv. Vennových diagramů







Všechny logické funkce 2 proměnných

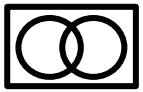
- Log. výrazy v disjunktní formě a pravdivostní tabulka

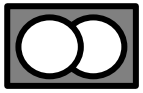
$$f_0 = 0$$
$$f_1 = (\bar{x}_1 \cdot \bar{x}_2)$$
$$f_2 = (x_1 \cdot \bar{x}_2)$$
$$f_3 = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2)$$
$$f_4 = (\bar{x}_1 \cdot x_2)$$
$$f_5 = (\bar{x}_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2)$$
$$f_6 = (x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2)$$
$$f_7 = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2)$$
$$f_8 = (x_1 \cdot x_2)$$
$$f_9 = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot x_2)$$
$$f_{10} = (x_1 \cdot \bar{x}_2 + x_1 \cdot x_2)$$
$$f_{11} = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 + x_1 \cdot x_2)$$
$$f_{12} = (\bar{x}_1 \cdot x_2 + x_1 \cdot x_2)$$
$$f_{13} = (\bar{x}_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 + x_1 \cdot x_2)$$
$$f_{14} = (x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 + x_1 \cdot x_2)$$
$$f_{15} = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 + x_1 \cdot x_2) = 1$$

x_2	x_1	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	**0**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	**1**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	**0**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	**1**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Všechny logické funkce 2 proměnných

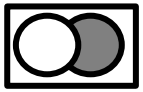
- Vennovy diagramy, logické funkce a operace

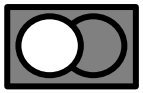

Kontradikce $f_0 = 0$

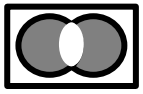

Pierceova funkce (NOR) $f_1 = (\bar{x}_1 \cdot \bar{x}_2) = \overline{x_1 + x_2} = (x_1 \downarrow x_2)$

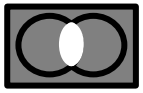

Inhibice $f_2 = (x_1 \cdot \bar{x}_2) = \overline{(x_1 \Rightarrow x_2)}$


Negace (NOT) $f_3 = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2) = \bar{x}_2$


Inhibice $f_4 = (\bar{x}_1 \cdot x_2) = \overline{(x_2 \Rightarrow x_1)}$

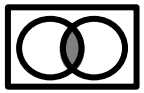

Negace (NOT) $f_5 = (\bar{x}_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2) = \bar{x}_1$

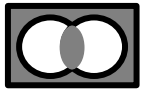

Exkluzívní součet (XOR) $f_6 = (x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2) = (x_1 \oplus x_2)$

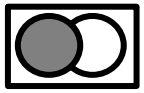

Shefferova fce (NAND) $f_7 = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2) = \overline{(x_1 \cdot x_2)} = (x_1 \uparrow x_2)$

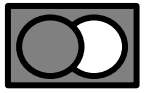
Všechny logické funkce 2 proměnných

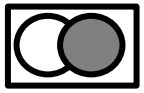
- Vennovy diagramy, logické funkce a operace


Log. součin (AND) $f_8 = (x_1 \cdot x_2)$

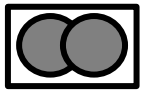

Ekvivalence (XNOR) $f_9 = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot x_2) = (x_1 \Leftrightarrow x_2)$


Identita $f_{10} = (x_1 \cdot \bar{x}_2 + x_1 \cdot x_2) = x_1$


Implikace $f_{11} = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 + x_1 \cdot x_2) = (x_1 + \bar{x}_2) = (x_2 \Rightarrow x_1)$


Identita $f_{12} = (\bar{x}_1 \cdot x_2 + x_1 \cdot x_2) = x_2$


Implikace $f_{13} = (\bar{x}_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 + x_1 \cdot x_2) = (\bar{x}_1 + x_2) = (x_1 \Rightarrow x_2)$


Log. součet (OR) $f_{14} = (x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 + x_1 \cdot x_2) = (x_1 + x_2)$


Tautologie $f_{15} = (\bar{x}_1 \cdot \bar{x}_2 + x_1 \cdot \bar{x}_2 + \bar{x}_1 \cdot x_2 + x_1 \cdot x_2) = 1$

Logické členy OR, AND a NOT

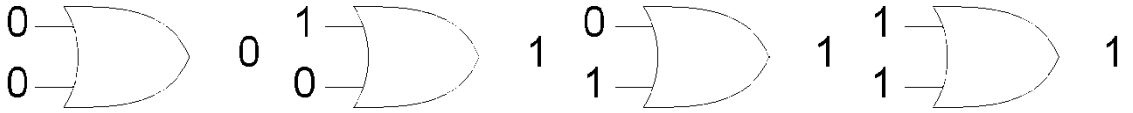
- Definiční tabulka
  - Logický součet

+	**0**	**1**
0	0	1
1	1	1

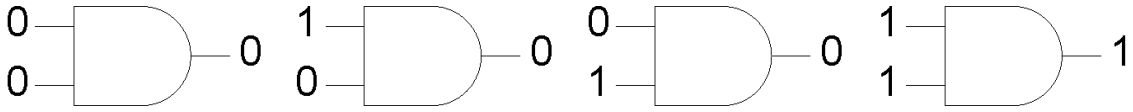
Logický součin

·	**0**	**1**
0	0	0
1	0	1

- Logické operace se realizují tzv. logickými členy
  - „+" … log. člen OR



  - „·" … log. člen AND



- Chování log. členů lze definovat např. pravdivostní tabulkou
  - OR

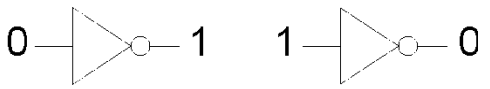
a	**b**	**a+b**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

AND

a	**b**	**a·b**
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

NOT

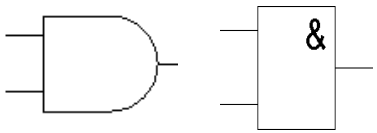
a	**not(a)**
0	1
1	0



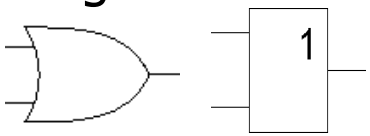
Logické členy

- Pomocí základních log. členů AND, OR a NOT lze realizovat libovolný logický obvod a tedy i číslicový systém (viz dále)
  - Log. funkce AND a OR jsou (s použitím log. funkce NOT) komplementární (lze je vhodným způsobem vzájemně nahradit)
  - Dokonce stačí použít log. členy NAND a NOR pouze se dvěma vstupy (NAND = AND s invertorem na výstupu, nebo NOR = OR s invertorem na výstupu)
- Značení
  - Čtvercové značky - funkce logického členu je označena znaky "&" pro funkci AND, "1" pro funkci OR
  - Značky složené z křivek - rozšířené ve většině profesionálních systémů pro návrh logických obvodů (každý způsob značení má své výhody a nevýhody)

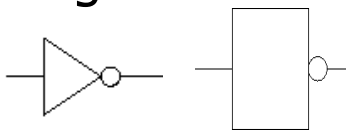
Log. člen AND



Log. člen OR

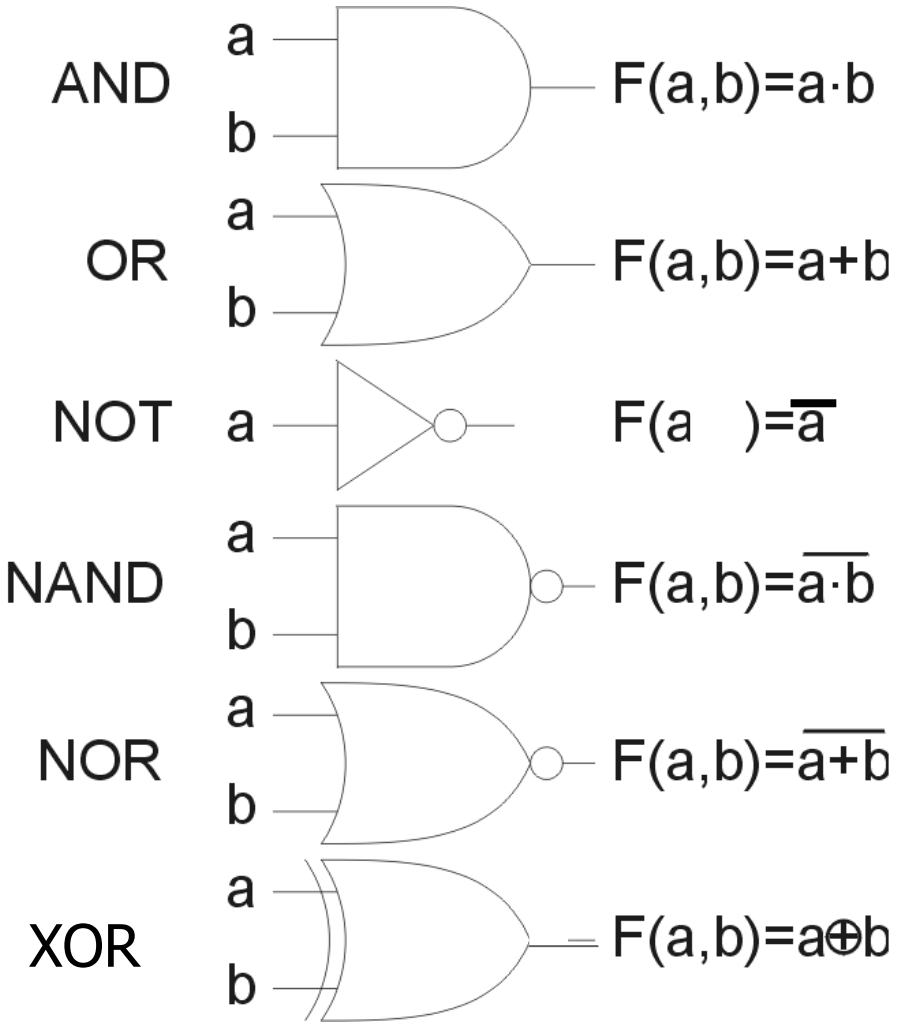


Log. člen NOT

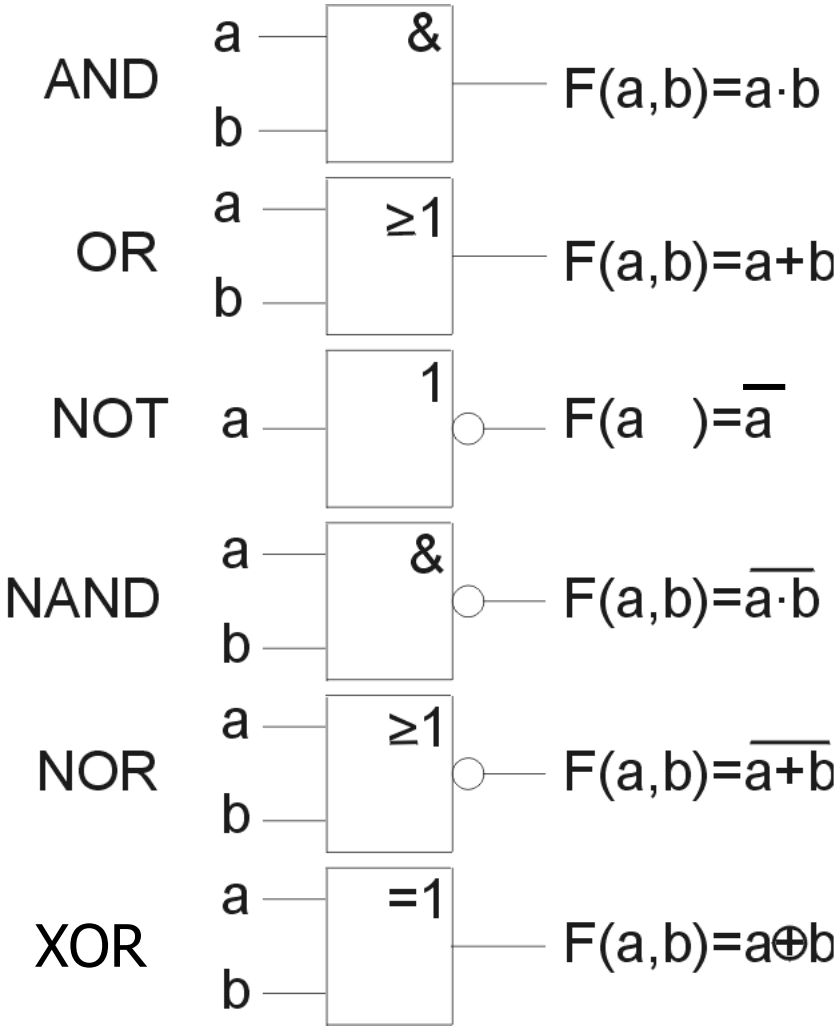


Logické členy: Schematické značky (symboly)

- Standard IEEE/ANSI Std 91-1984



- Standard IEC 60617-12

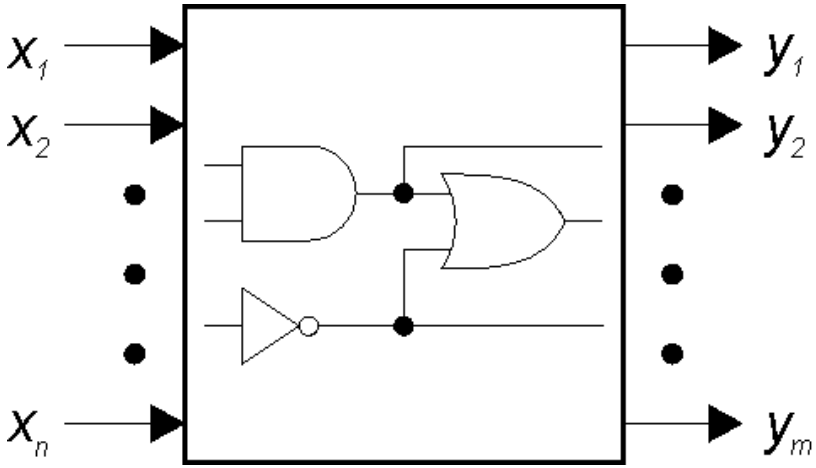


Logické obvody

- Číslicové systémy
  - Sestavujeme z obvodů, které navrhujeme za použití Booelovy algebry (práce s logickými výrazy)
  - Proto těmto obvodům říkáme též logické obvody
- Logické obvody dělíme do dvou skupin dle jejich chování
  - Kombinační logické obvody
  - Sekvenční logické obvody
- Kombinační i sekvenční logické obvody
  - Skládají se ze stejných elementárních prvků, tzv. logických členů
- Logické členy
  - Logické členy mají jeden či více vstupů a jeden výstup
  - Hodnota na výstupu log. členu je funkcí hodnot vstupních
    - Log. členy se též nazývají "hradla" (anglicky gate)

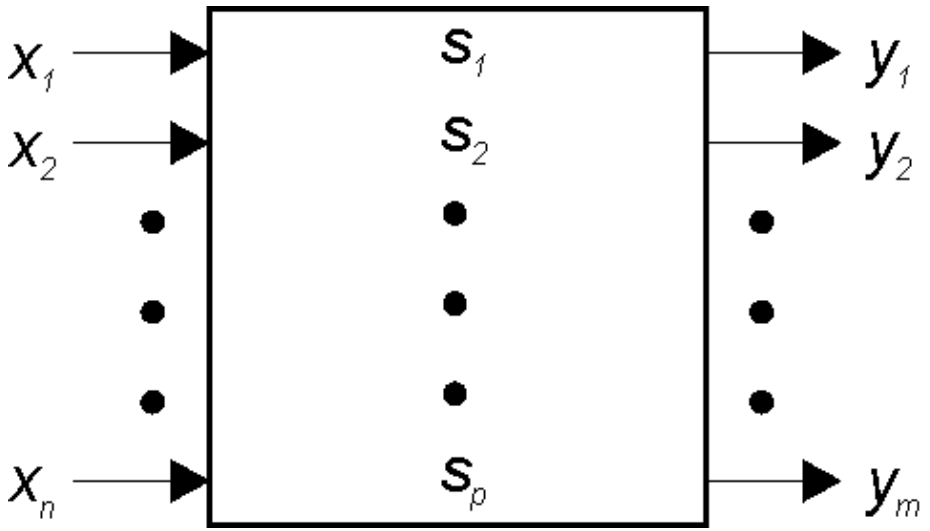
Kombinační logické obvody

- Hierarchicky uspořádaná struktura, ve které jednotlivé komponenty zpracovávají a mezi sebou přenáší informaci reprezentované v binární formě (log. úrovně)
  - Každá komponenta má kombinační chování
  - Vstup každé komponenty je připojen pouze k jednomu výstupu předchozí komponenty nebo ke zdroji log. "0" či "1" (nelze spojovat výstupy - není jasné, který výstup je platný)
    - Pozn.: tzv. montážní logické členy tuto podmínku neporušují, viz dále
  - Struktura neobsahuje cykly (zpětné vazby)
  - Funkční a časové chování lze odvodit z funkčního a časového chování jednotlivých komponent



Sekvenční logické obvody

- Hodnoty výstupních proměnných jsou závislé nejen na aktuální kombinaci hodnot vstupních proměnných, ale též na předchozích hodnotách vstupních proměnných a počátečním stavu
  - Říkáme, že sekvenční obvody mají "paměť"
  - Praktická realizace paměti stavu může být různá
- Aktuální hodnoty stavových proměnných
  - Uchovávají veškerou informaci o minulosti, která je potřebná pro stanovení budoucího chování obvodu



Kvantifikace informace (C. Shannon, 1948)

- Entropie (míra neurčitosti) vs. informace (míra určitosti)
  - S rostoucí mírou informace klesá míra entropie
  - Nejjednodušší případ je situace, kdy volíme ze dvou možností
  - Pokud se dozvíme, že jedna ze dvou možností platí, dostaneme elementární množství informace - bit (binary digit – binární číslo)
  - Binární informace je měřena v bitech (b) - počet jedniček nebo nul potřebných pro zakódování daných N možností
  - Množství informace - pokud máme N možností a nějaký fakt zúží počet možností na M, pak platí, že množství informace $I = \log_2(N/M)$ [bitů]
- Příklad
  - Hod mincí: $I_1 = \log_2(2/1) = 1$ b
  - Hod dvěma kostkami: $I_2 = \log_2(6 \cdot 6/1) = 5{,}2$ b

Kódování informace

- Kódování = přidělení jisté reprezentace konkrétní informaci
  - Volba kódu výrazně ovlivňuje vlastnosti implementace příslušného systému (netriviální úloha)
- Problematika zahrnuje
  - Počet potřebných bitů (množství komponent – cena a spolehlivost systému)
  - Rychlost manipulace s bity (výkonnost systému)
  - Energetické nároky (změny hodnot odebírají nejvíce energie)
  - Délka (fixní, proměnná)
  - Čísla (bez a se znaménkem, pevná a plovoucí řádová čárka)
  - Komprese (ztrátová, bezztrátová)
  - Šifrování, autorizace
  - Detekce, oprava chyb (redundance, dostupnost)
  - Odolnost proti rušení atd.

Fyzická reprezentace binární informace

- Reprezentace
  - Abstraktní forma - pouze hodnoty 0 a 1
  - Reálná forma - "nízká" hodnota = 0, "vysoká" hodnota = 1
- Běžně dostupné technologie - elektronické obvody
  - Pracují s elektrickými veličinami - napětí, fáze, proud atd.
  - Elektrické napětí - dnes dominuje (lze snadno generovat i měřit, existuje dlouhodobá zkušenost)
- Hodnota napětí je ovlivněna
  - Nepřesnostmi při jeho generování i měření (reálný svět není diskrétní, ale spojitý)
  - Rušením, výrobními tolerancemi, prostředím (teplota) apod.
- Aby reálný číslicový systém pracoval spolehlivě, musí:
  - Tolerovat určitou chybu ("nízká" a "vysoká" hodnota)
  - Se chovat jako by byl diskrétní (abstrakce 0 a 1)

Reprezentace binární informace: Napětí

- Logické úrovně (pozitivní logika)
  - "Nízká", nula - hodnota napětí reprezentující abstraktní "0"
  - "Vysoká", jednička - hodnota nap. reprezentující abstraktní "1"
- Fyzická reprezentace logických úrovní
  - Obvody, které spolehlivě (nelze dodržet absolutně) zajistí, že není možno (za normálních podmínek) zaměnit "0" za "1"
- Realizace
  - Obvod má definován "ochranný" interval napětí mezi hodnotami reprezentujícími "0" a "1" (tzv. zakázaná oblast)
  - Napětí v zakázané oblasti není reprezentováno ani jako "0", ani jako "1" = nejsou zde definovány platné logické úrovně
  - Příklad:

Komunikace binární informace: Rušení

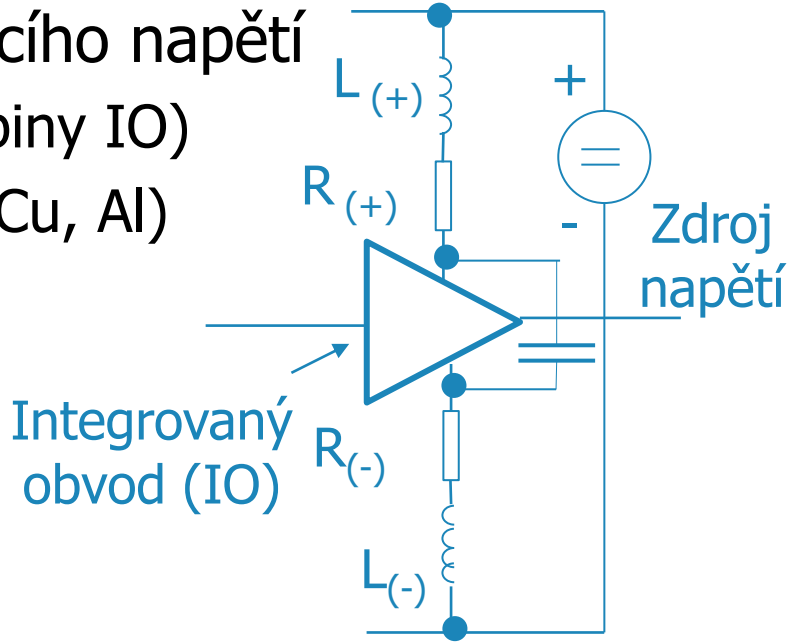
- Informace (logické úrovně) se v číslicovém systému přenáší vodiči
  - Ve vodiči se může indukovat rušivé napětí, které se přičítá k napětí odpovídajícímu platné logické hodnotě => hodnota napětí se může dostat do zakázané oblasti a obvod pak nepracuje tak, jak je očekáváno
- Číslicové obvody proto musí být navrženy tak, aby odolávaly rušení z různých vnějších zdrojů
  - Hovoříme o tzv. elektromagnetické kompatibilitě – jedna ze základních vlastností všech elektronických zařízení
- Elektronická zařízení musí být certifikována tak, aby platilo
  - Nelze jej „zarušit" - pracuje správně i při povolené úrovni elektromagnetického rušení daného pracovního prostředí
  - Samo neruší ostatní zařízení - nevyzařuje vyšší úrovně elektromagnetického rušení, než je povoleno

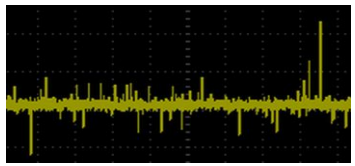
Rušení: Odrazy na vedení, odběr proudu

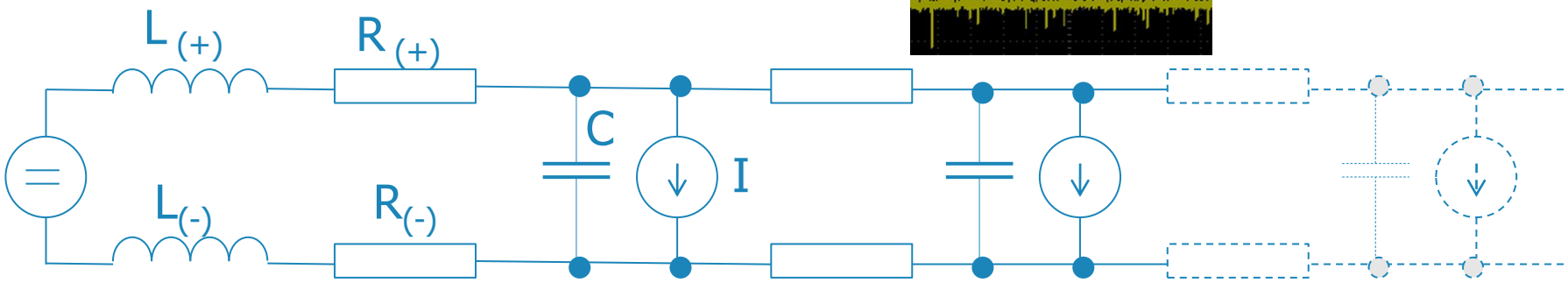
- Odrazy na vedení (vodičích)
  - Při šíření signálů dochází na konci vedení k jejich odrazům a zpětnému šíření k jejich zdroji
    - Elektrický signál se ve vodičích šíří konečnou rychlostí (shora omezeno rychlostí světla)
  - Odražený signál se přičítá k aktuální hodnotě na vedení – ovlivňování logických úrovní
  - Vedení je třeba tzv. impedančně přizpůsobit
- Změny odběru proudu obvodem (oproti ustáleným hodnotám)
  - Při změnách logických úrovní (přechody z 0->1 a 1->0)
  - Toto se projeví (díky úbytkům na napájecích rozvodech) změnami hodnot logických úrovní na výstupu
  - Nutno "odfiltrovat" pomocí blokovacích kondenzátorů na rozvodech napájecího napětí

Rušení: Rozvody napájecího napětí

- Náhradní schéma rozvodů napájecího napětí
  - L – indukčnost (přívodní vodiče, piny IO)
  - R – odpor propojovacích vodičů (Cu, Al)
  - C – parazitní kapacity mezi vodiči
  - I – odebíraný proud obvodu
- Zdroje rušivého napětí
  - $U_R = I \cdot R$, $U_L = di/dt$
  - LRC tvoří rezonanční obvod
  - Zvlnění napájecího napětí (ripple) je způsobeno poklesy napětí na R a nabíjením a vybíjením LC

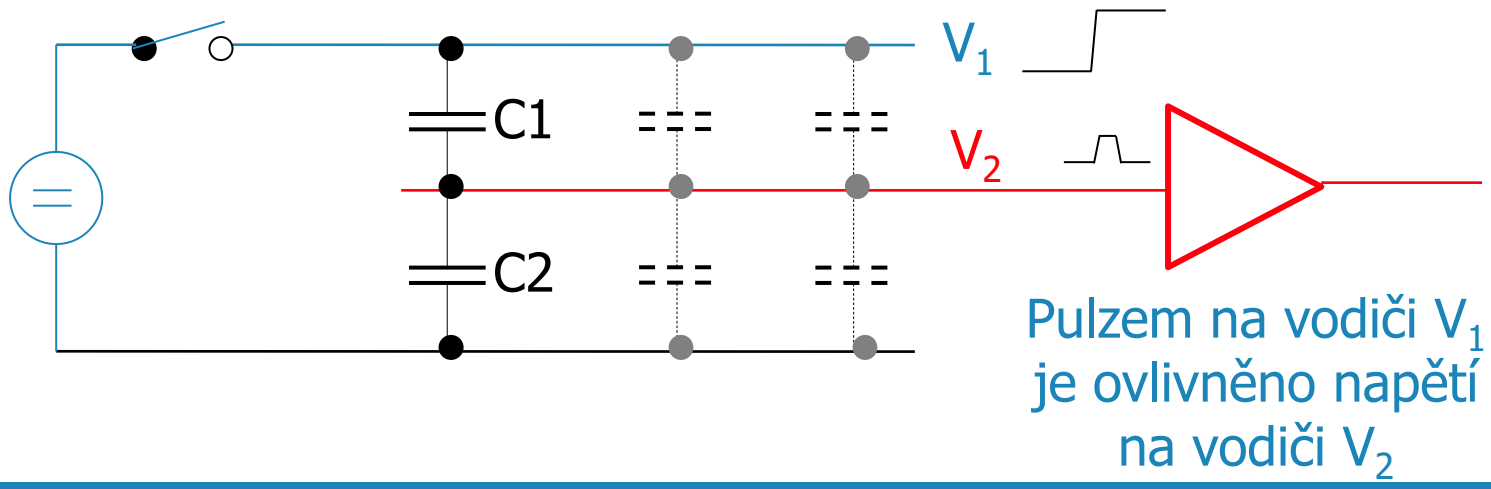






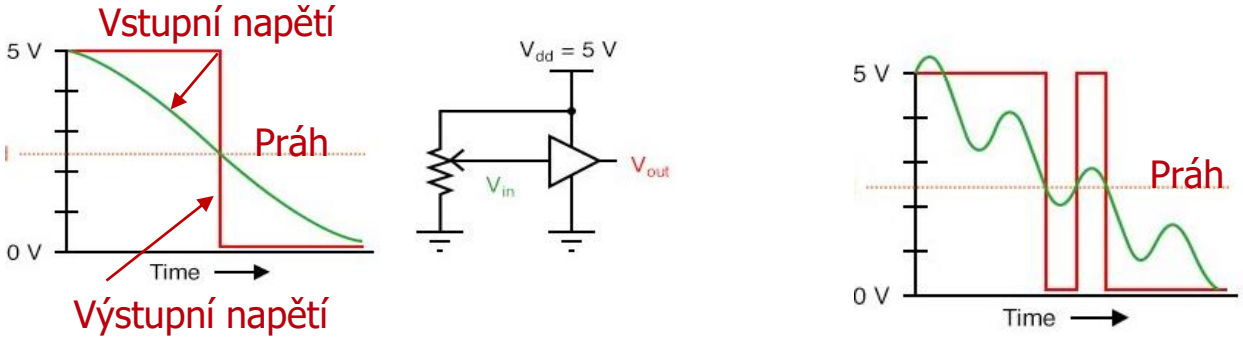
Rušení: Přeslechy

- Mezi dvěma blízkými vodiči dochází vlivem parazitních kapacit k přeslechům (crosstalk)
  - Pokud se na vodiči V_1 skokově změní napětí ΔU_1 (např. změna hodnoty napětí z logické 0 do logické 1), pak se na vodiči V_2 projeví změna napětí
  - $\Delta U_2 = C1/(C1+C2) \cdot \Delta U_1$
- Lze omezit pečlivým vedením vodičů, ne však plně eliminovat



Rušení: Příklad

- Díky rušení se na spoje indukuje napětí, které se přičítá k užitečnému signálu (logickým úrovním)
  - Pokud by logický obvod pracoval tak, že překmit nad a pod rozhodovací (prahové) napětí by generoval pulsy na výstupu, nebyl by v případě přítomnosti rušení použitelný (jednalo by so o analogový komparátor)

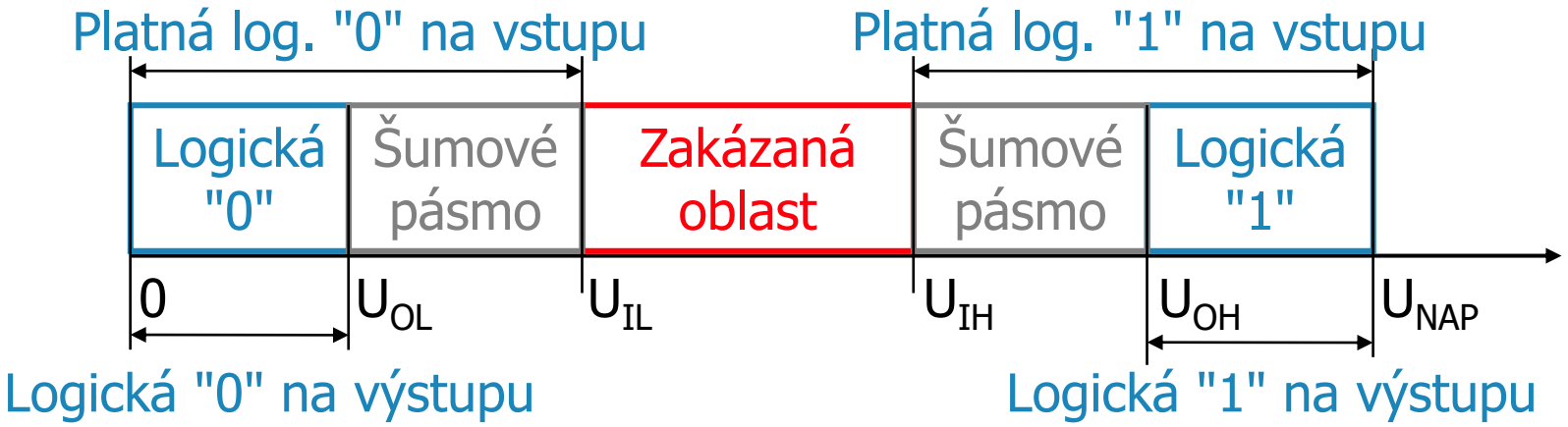


- Řešení - logické obvody "vylepšují" hodnoty napěťových úrovní mezi vstupem a výstupem
  - Akceptují „horší" a generuje "lepší" logické úrovně
  - Díky díky tomu pracují správně i při jisté úrovni rušení

Zdroj: https://www.allaboutcircuits.com/textbook/digital/chpt-3/logic-signal-voltage-levels/

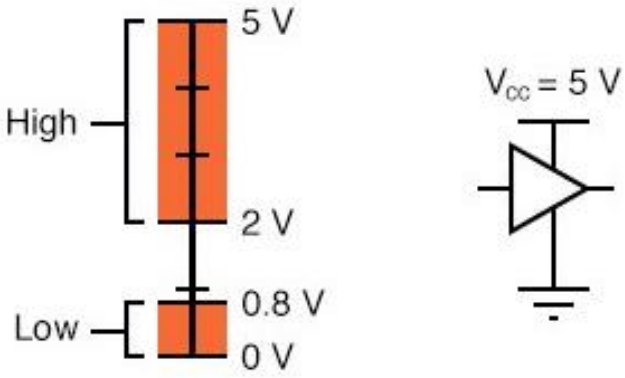
Rušení: Omezení vlivu konstrukcí logických obvodů

- Logická 0
  - Obvod na vstupu akceptuje napětí v rozsahu 0 až U_{IL}
  - Obvod výstupu generuje napětí v rozsahu 0 až U_{OL}
  - Obvod má pásmo akceptovatelného šumu U_{IL}-U_{OL}
- Logická 1
  - Obvod na vstupu akceptuje napětí v rozsahu U_{IH} až U_{NAP}
  - Obvod výstupu generuje napětí v rozsahu U_{OH} až U_{NAP}
  - Obvod má pásmo akceptovatelného šumu U_{OH}-U_{IH}

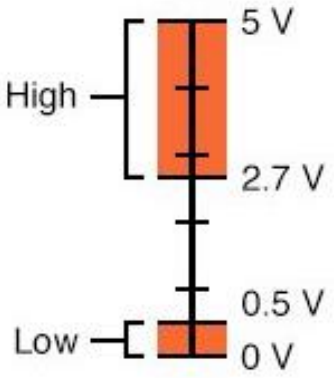


Rušení: Šumová odolnost – příklad technologie TTL

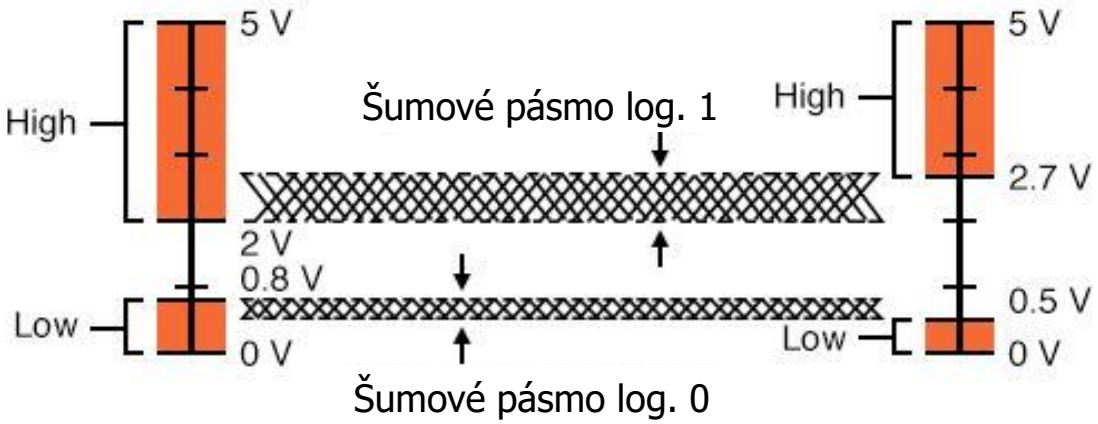
- Akceptované vstupní logické úrovně



- Generované výstupní logické úrovně



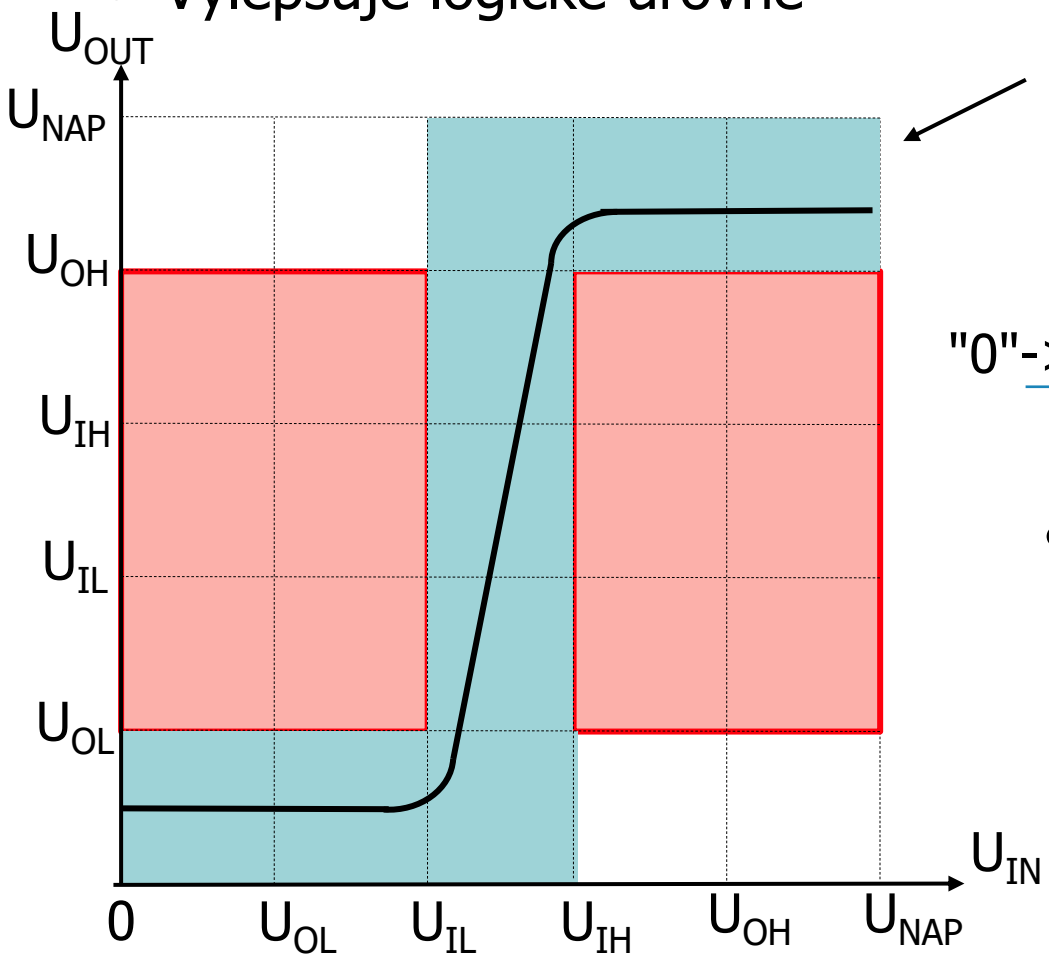
- Pásmo odolnosti proti rušení



[Zdroj: https://www.allaboutcircuits.com/textbook/digital/chpt-3/logic-signal-voltage-levels/]

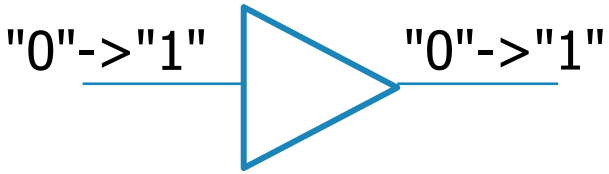
Obvod odolný proti rušení: Příklad

- Sledovač (tzv. buffer)
  - Přenáší (kopíruje) logickou hodnotu ze vstupu na výstup
  - Vylepšuje logické úrovně



Převodní charakteristika

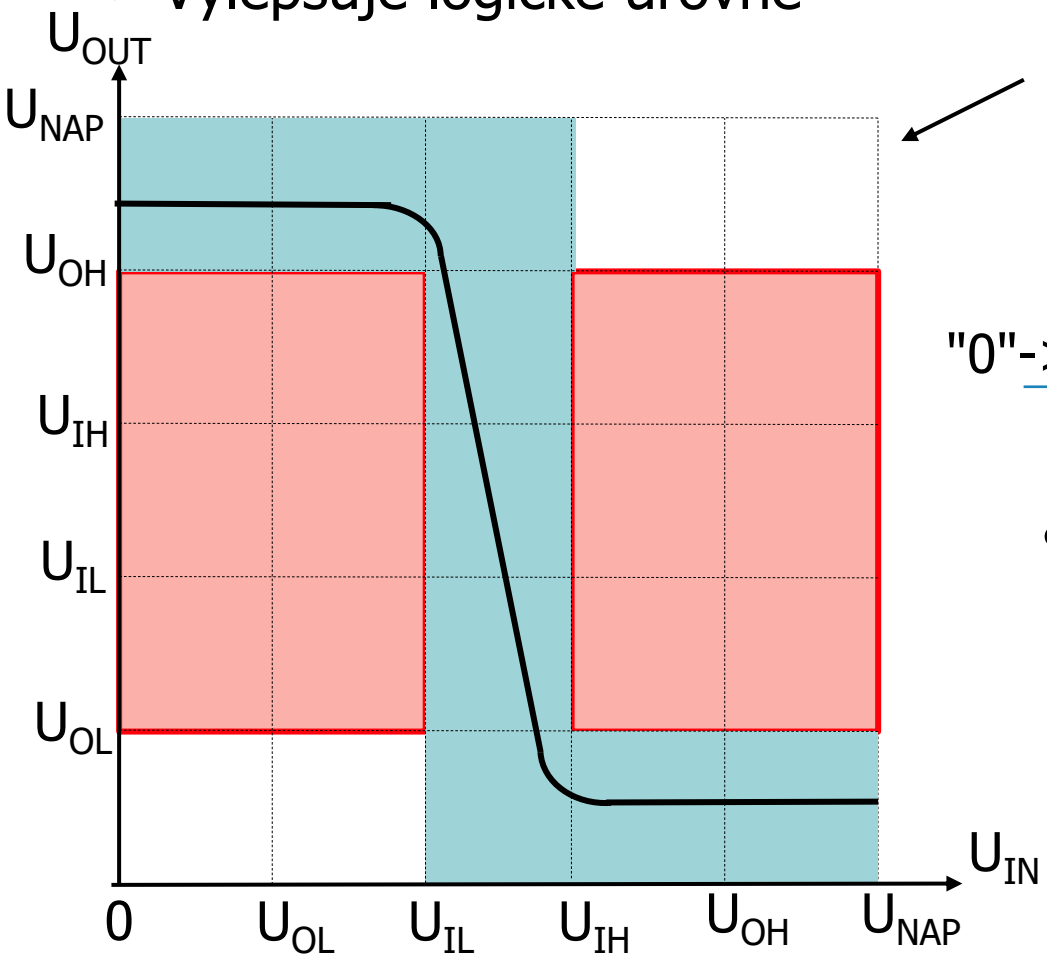
Symbol



- Obvod
  - Má zesílení A>1
  - Má zpoždění t_d>0

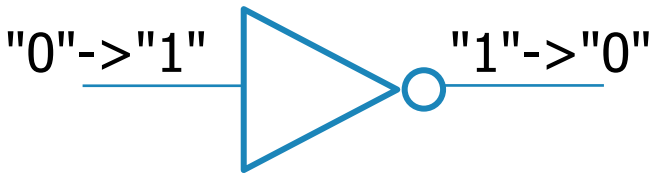
Obvod odolný proti rušení: Příklad

- Invertor
  - Invertuje (neguje) vstupní logickou hodnotu
  - Vylepšuje logické úrovně



Převodní charakteristika

Symbol

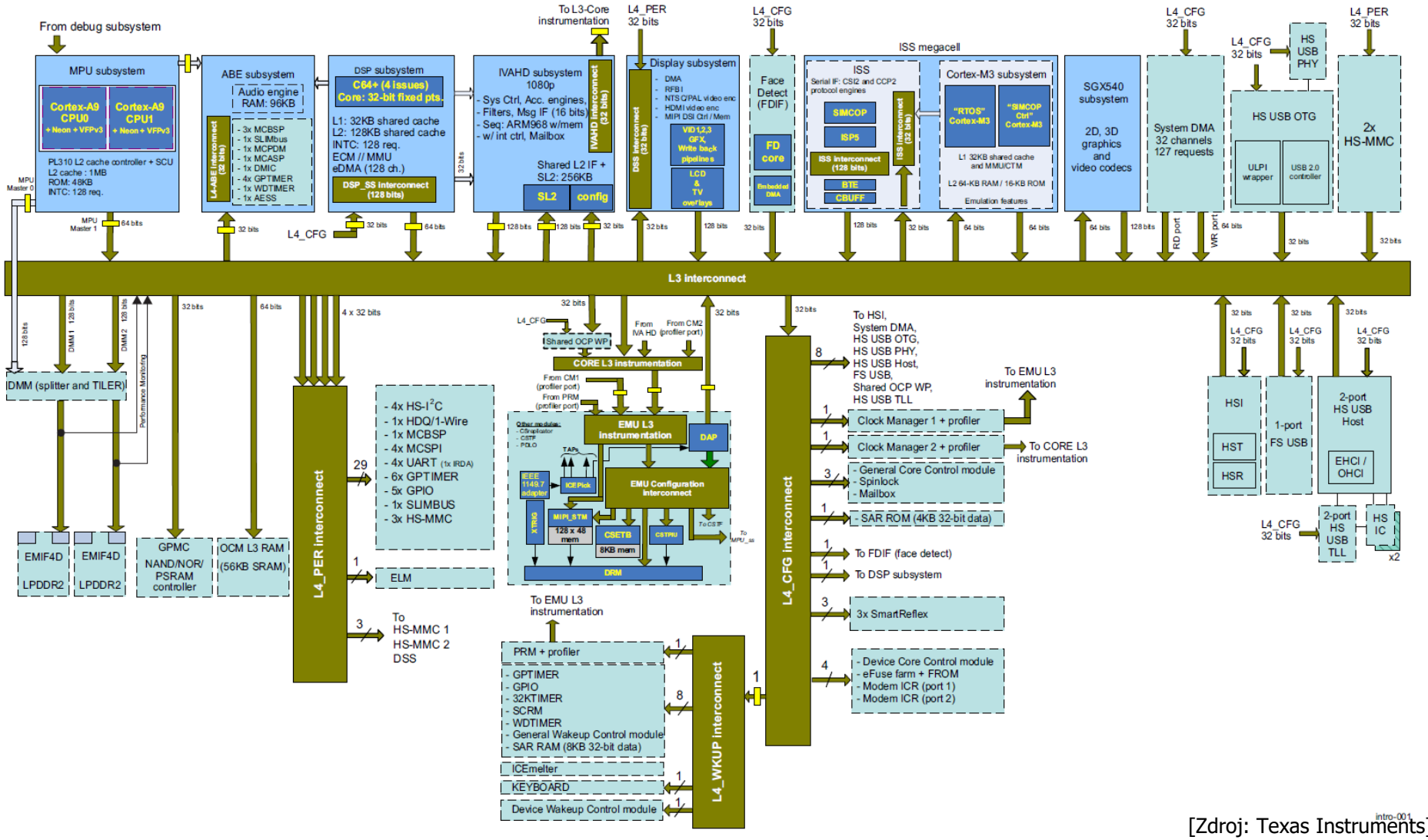


- Obvod
  - Má zesílení A>1
  - Má zpoždění t_d>0

Elektrické vlastnosti log. členů

- Logické úrovně
  - Log. členy jsou konstruovány tak, aby byly za normálních podmínek (teplota, napájecí napětí, rušení atd.) schopny generovat log. úrovně v jistém intervalu hodnot a též rozlišovat log. úrovně v určitém rozmezí hodnot
- Odolnost proti rušení (DC Noise Margins)
  - Je zajištěna v určitém rozmezí tak, že log. člen je schopen akceptovat větší rozptyl vstupních hodnot log. úrovní, než jaký generuje na výstupu
  - Rušení může být generováno např. kosmickým zářením, elektromagnetickým polem, kolísáním napájecího napětí apod.
- Logický zisk (Fan-Out)
  - Počet vstupů log. členů, které můžeme zapojit na výstup daného členu, při kterém jsou ještě zaručeny správné hodnoty log. úrovní pro celý rozsah pracovních podmínek (napájecí napětí, teplota)
- Rychlost
  - Doba, které je třeba k přechodu signálu ze vstupu na výstup
  - Dána dobami přechodů mezi log. úrovněmi a dobou průchodu signálu
  - Závisí na konstrukci log. členů, na počtu jiných log. členů zapojených na jeho výstup, na délce vodičů, na konstrukci desky s plošnými spoji atd.
- Příkon
  - Závisí na konstrukci log. členu, na počtu členů, frekvenci změn log. úrovní, parazitních atd.

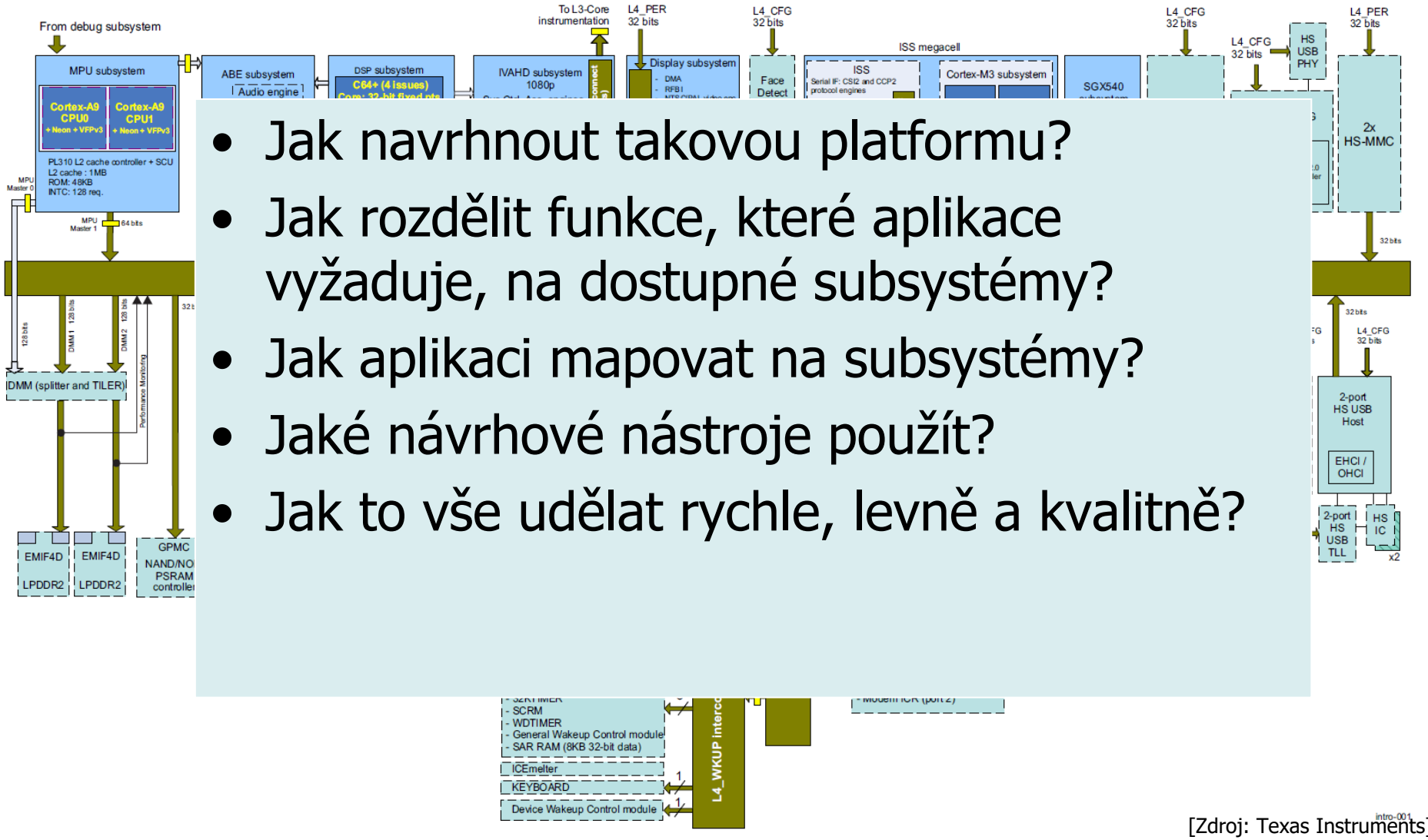
- Platforma OMAP4430 - SoC pro mobilní zařízení



[Zdroj: Texas Instruments]

Limity: Složitost

- Platforma OMAP4430 - SoC pro mobilní zařízení



- Jak navrhnout takovou platformu?
- Jak rozdělit funkce, které aplikace vyžaduje, na dostupné subsystémy?
- Jak aplikaci mapovat na subsystémy?
- Jaké návrhové nástroje použít?
- Jak to vše udělat rychle, levně a kvalitně?

[Zdroj: Texas Instruments]

Návrh složitých systémů

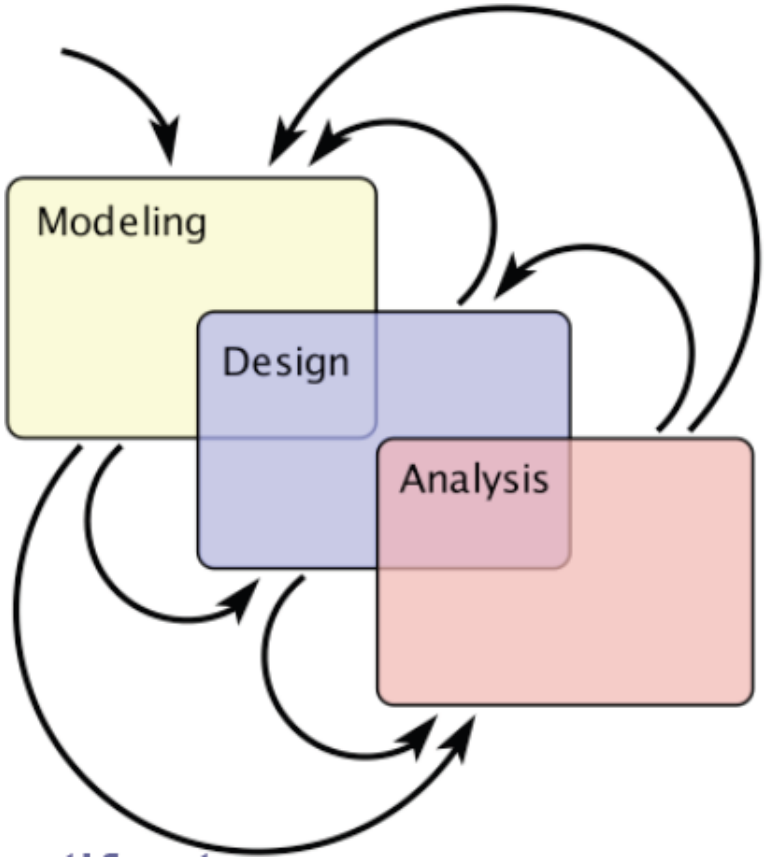
- Složité systémy
  - Jsou sestaveny z jednodušších komponent, které vzájemně komunikují a kooperují a společně utváří chování či vlastnost systému, jež jde nad rámec možností jednotlivých komponent
- Vědecké metody, inženýrský přístup, umění
  - Modelování, optimalizace, verifikace…
  - Vzdělání, zkušenosti, tradice, koncepce…
  - Invence, kreativita, estetika, vize, intuice…
- Kompromisy
  - Hledání kompromisů mezi potřebami uživatelů a možnostmi technologií (cena, výkonnost, spotřeba energie atd.)
- Prostředí
  - Je třeba respektovat životní prostředí a aspekty sociální, etické, zdravotní, bezpečnostní, výrobní, servisní a další

Návrh složitých systémů

- Techniky návrhu
  - Dekompozice (struktura = komponenty + rozhraní)
  - Hierarchie (snaha o co nejvyšší úroveň abstrakce při návrhu)
  - Vhodná specifikace (popis struktury a chování)
  - Způsob popisu (shora-dolů, zdola-nahoru)
  - Automatizace návrhu (syntéza HW, kompilace SW)
  - Validace (testování, verifikace,…)
- Dekompozice systému a zavedení hierarchie vyžaduje definici rozhraní mezi subsystémy => rozhraní:
  - Umožňuje dekompozici systému a zavedení hierarchie
  - Izoluje jednotlivé subsystémy (komponenty) a různé technologie mezi sebou a umožňuje jejich vzájemné propojení
  - Umožňuje stavbu systému ze standardních komponent
  - Má často delší životnost než samotný systém (viz např. USB)

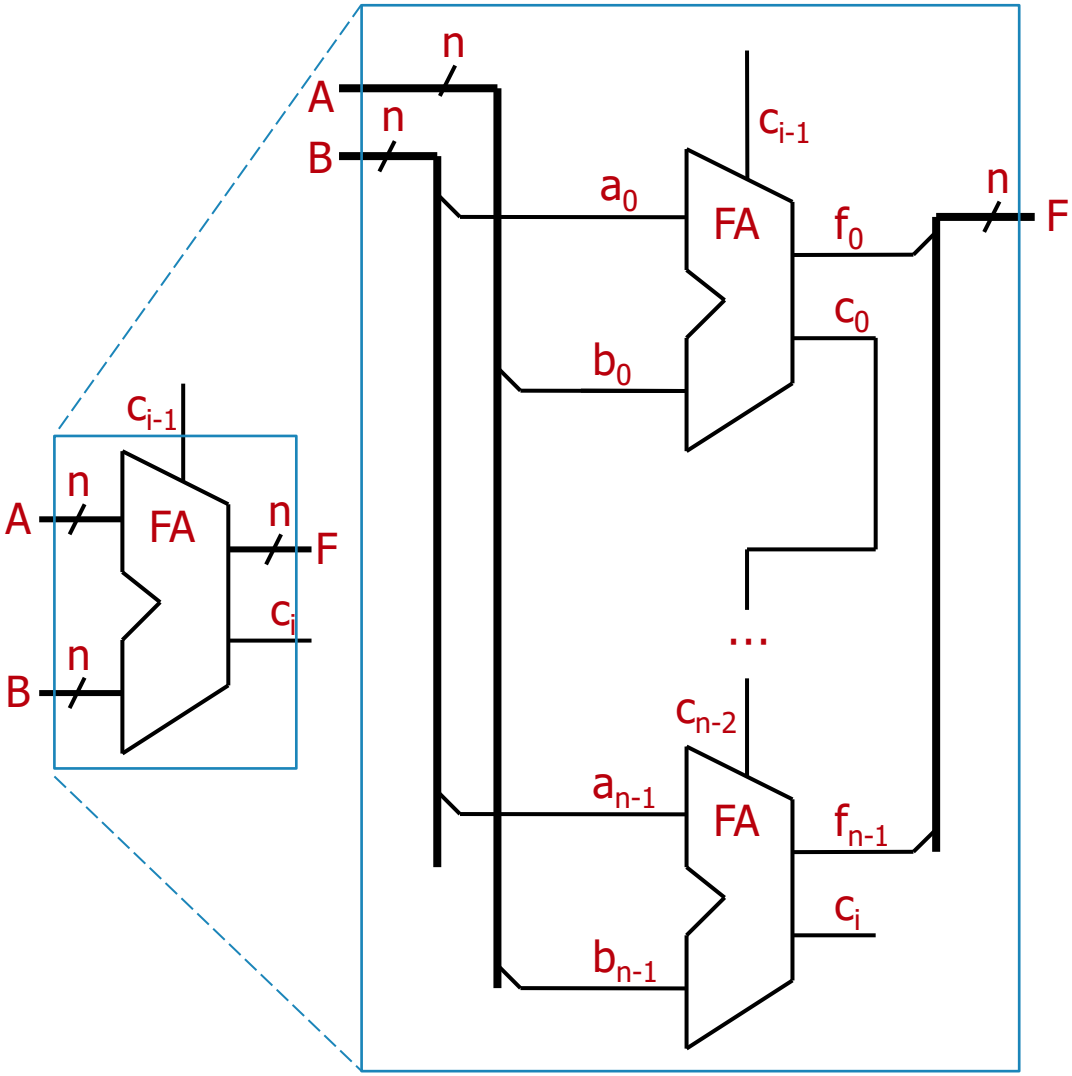
Modelování, návrh, analýza

- Modelování – specifikuje, co systém dělá
  - Proces získávání hlubší znalosti o systému jeho imitací
- Návrh (design) – specifikuje, jak systém pracuje
  - Strukturovaná tvorba výrobku
- Analýza – specifikuje, proč systém dělá to, co dělá
  - Proces získávání hlubší znalosti o systému jeho podrobným zkoumáním



Popis systémů: Hierarchie

- Pro zjednodušení návrhu je výhodné používat hierarchický popis systémů
- Hierarchie zapouzdřuje (seskupuje) jednodušší obvody do složitějších celků (komponent), se kterými se lépe pracuje
- Příklad:
  N jednobitových úplných sčítaček (FA) je seskupeno do jedné N-bitové úplné sčítačky

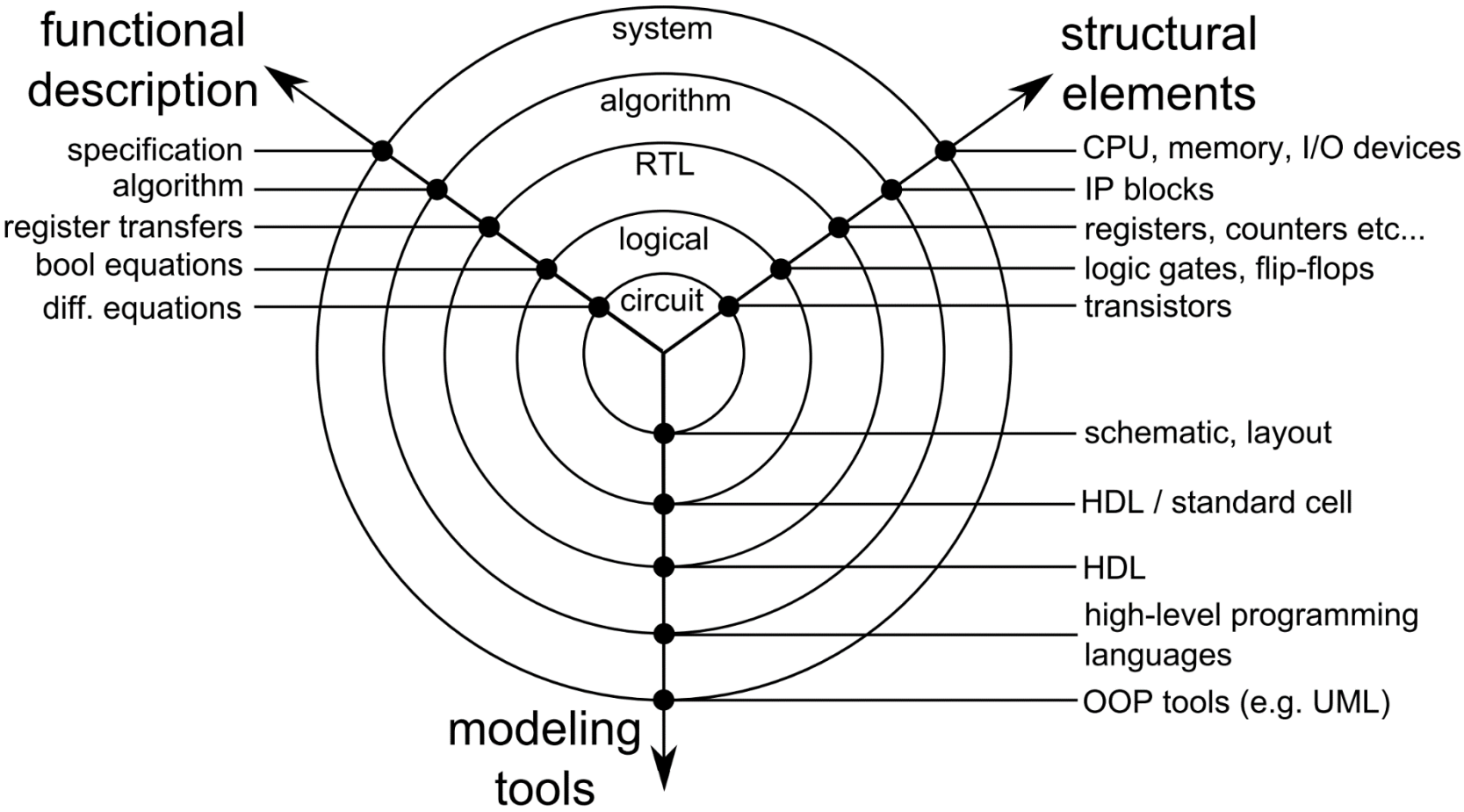


Popis systémů: chování - struktura

- Popis chování
  - Zápis algoritmu, který má obvod vykonávat
    - Matematický výraz
    - Programovací jazyk
    - Vývojový diagram
    - Tabulka
    - Časový diagram, atd.
- Specifikace toho, co má systém dělat (ne toho, jak bude implementován)
  - Tvůrčí činnost, kterou lze jen omezeně automatizovat
- Popis struktury
  - Definice a propojení jednotlivých prvků systému
    - Schéma
    - Programovací jazyk, atd.
- Specifikace toho, jak bude systém implementován (ne toho, co bude dělat)
  - Jednotlivé komponenty, ze kterých se systém skládá
  - Všechny signály, kterými se přenáší informace jak mezi komponentami, tak okolím
  - Tok informace v systému
  - Lze automatizovat

Popis systémů: Úrovně abstrakce

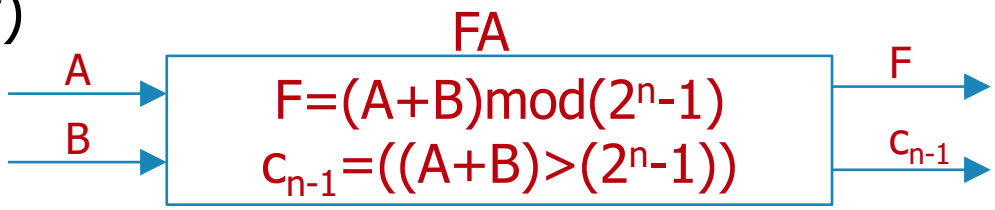
- Popis chování (behaviorální, funkční), popis struktury, modelovací nástroje

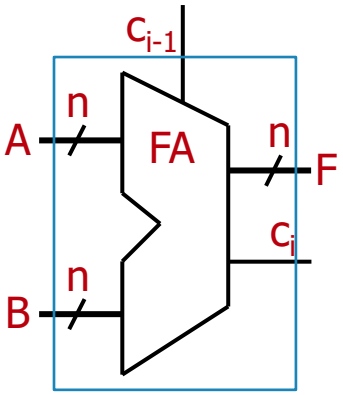


[Obrázek: http://www.eet.bme.hu/~horvathp/contents/aramkortervezes/eloadasok/01_Abstraction_Levels_in_the_Digital_System_Modeling.pdf]

Popis systémů: Úrovně abstrakce

- Nižší úrovně abstrakce nesou více informací o struktuře výsledného obvodu, vyšší popisují jeho chování
  - Přechody z vyšší do nižší úrovně abstrakce = kroky návrhu
- Příklad – úplná N-bitová sčítačka složena z 1bitových
  - Popsána algebraickými výrazy (popis chování) a jako logická síť (popis struktury)



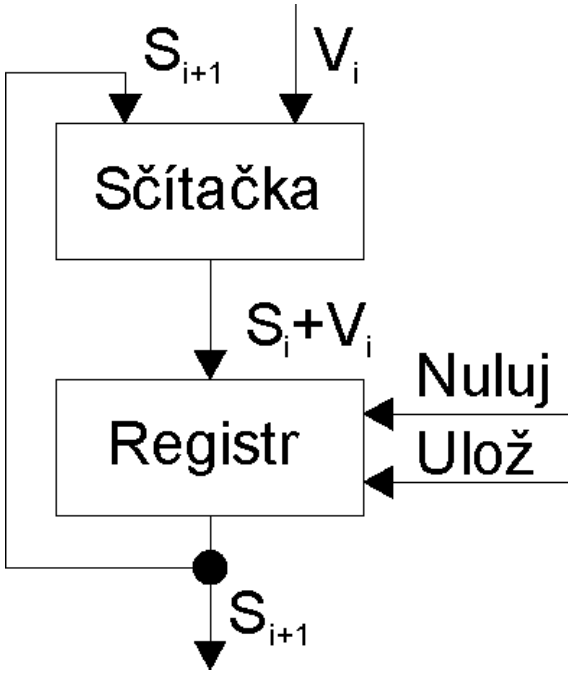


Úroveň popisu	Úroveň abstrakce	Množství detailů
Systém	Nejvyšší	Nejméně
Přenosy mezi registry	↑	↓
Log. obvody a členy	↑	↓
Tranzistory	↑	↓
Polovodičová podložka	Nejnižší	Nejvíce

Úroveň popisu: Přenosy dat mezi registry (RTL)

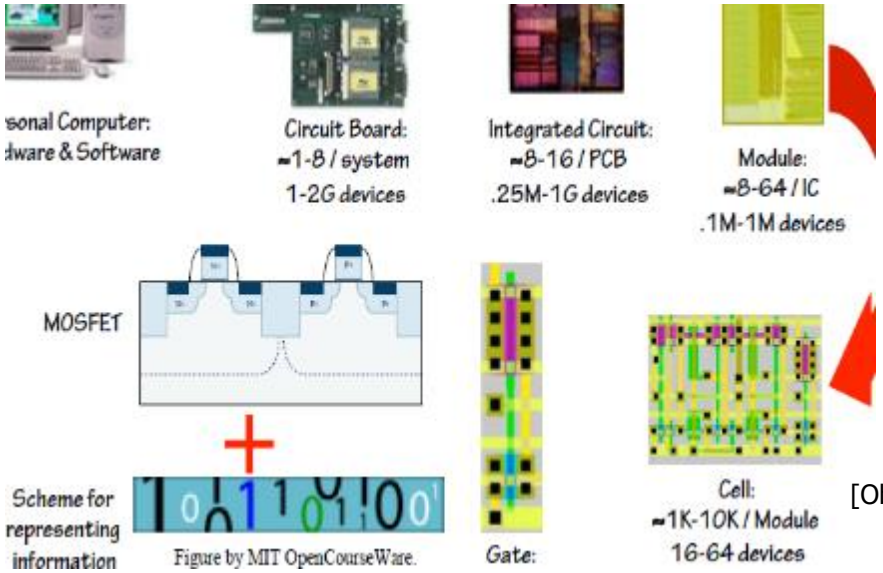
- Příklad - součet členů posloupnosti
- Popis chování - programovacím jazykem
- Popis struktury - graficky schématem
  - Funkční jednotka – sčítačka, provádí součet dvou čísel (mezivýsledek plus další člen posloupnosti)
  - Registr S (paměť) uchovává (akumuluje) jednotlivé mezivýsledky
  - Sčítačka a registr jsou propojeny signály
  - Registr je na počátku nejprve signálem *Nuluj* vynulován a dále si postupně, na povel signálu *Ulož*, pamatuje novou hodnotu výsledku

$$S_0 \leftarrow 0; \qquad vynulujregistr$$
$$\underline{for}\ i = 1\ \underline{to}\ N\ \underline{do}; \qquad provšechna\ V_i$$
$$(S_{i+1} \leftarrow S_i + V_i); \qquad akumuluj$$

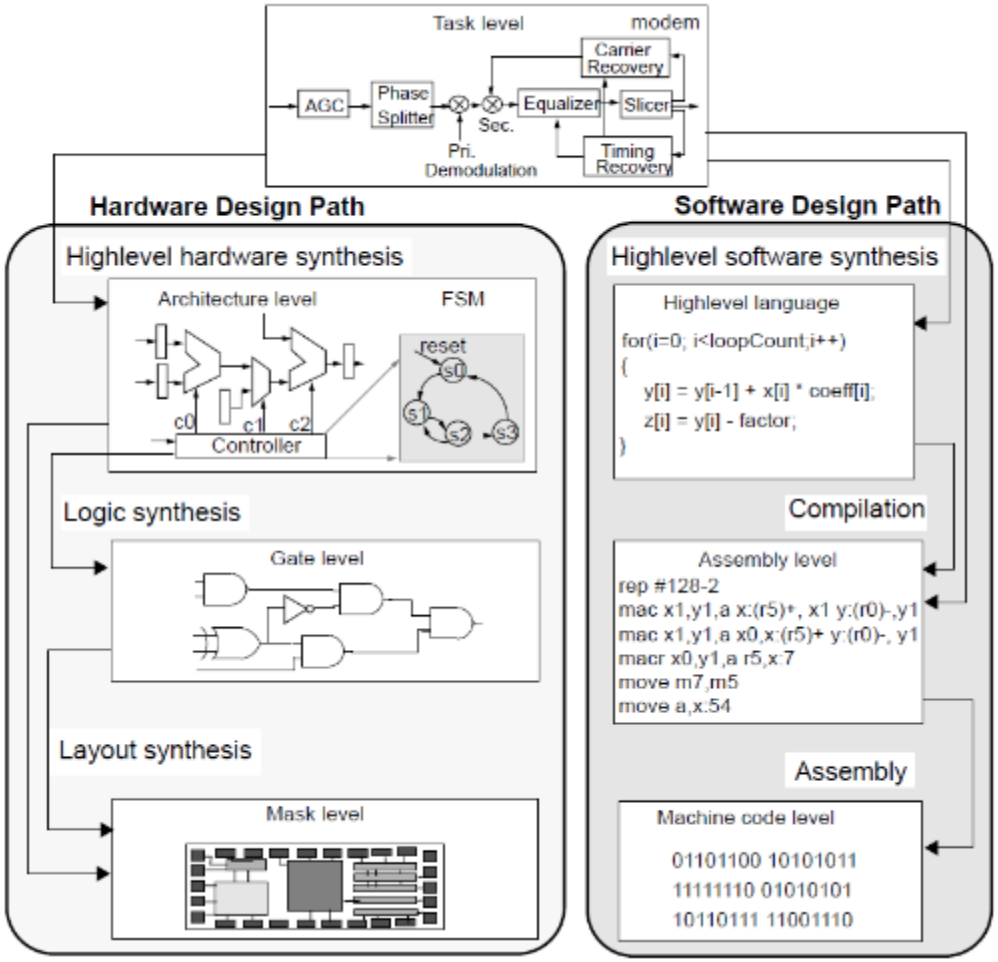


Různé úrovně a způsoby popisu systému: Příklady

- Osobní počítač
- Deska s plošnými spoji
- Integrovaný obvod
- Komponenta
- Logická buňka
- Hradlo
- Tranzistor
- Reprezentace informace (0, 1)



- Modem – SW a HW



[Obrázek: A. P. Kalavade: System-Level Codesign of Mixed Hardware-Software Systems, University of California, Berkeley, Technical Report No. UCB/ERL M95/88, 1995]

Postup při návrhu číslicových systémů

- Shora-dolů (anglicky top-down)
  - Nejprve definujeme chování systému a postupně konkretizujeme jednotlivé části – od obecného po podrobný popis struktury (podklad pro výrobu)
- Zdola-nahoru (anglicky bottom-up)
  - Vycházíme z komponent, které máme k dispozici
  - Postupně z nich skládáme jednotlivé bloky, z nichž postupně vybudujeme celý systém
- Oba přístupy je nezbytné vhodně kombinovat
  - Při návrhu algoritmu pro řešení daného problému je snahou postupovat co nejvíce „shora-dolů"
  - V praxi je však třeba též postupovat „zdola-nahoru" – pro optimalizaci návrhu (cena, příkon apod.) je třeba znát cílovou technologii, ve které bude systém fyzicky realizován, a využívat její komponenty

Návrh = algoritmus + syntéza

- Návrh = tvorba algoritmu a jeho implementace (syntéza)
- Algoritmus
  - Intuitivně - postup, který nás dovede k řešení úlohy
  - Formálně – „Přesně definovaná konečná posloupnost příkazů (kroků), jejichž prováděním pro každé přípustné vstupní hodnoty získáme po konečném počtu kroků odpovídající výstupní hodnoty“ [z kurzu Základy programování]
- Syntéza
  - Z formálního popisu algoritmu je třeba vytvořit výpočetní strukturu (logický obvod), která jej implementuje
  - Tvorba vhodné struktury se dnes provádí do značné míry automatizovaně
- Existuje mnoho výpočetních struktur, které implementují daný algoritmus (vícenásobná realizace)
  - Jejich vlastnosti určují rychlost výpočtu, příkon, rozměry, atd.