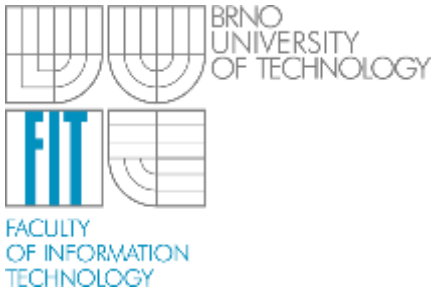
Návrh číslicových systémů (INC)

Otto Fučík

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



Použitá literatura

N. Frištacký, M. Kolesár, J. Kolenička a J. Hlavatý: "Logické systémy", SNTL Praha, 1986
M. Eysselt: "Logické systémy", SNTL Praha, skriptum VUT v Brně, 1985
J. F. Wakerly: "Digital Design. Principles and Practices", Prentice Hall, ISBN 0-13-769191-2, 2000
V. P. Nelson, H.T.Nagle, B.D.Carroll, J.D.Irwin: "Digital Logic Circuit Analysis & Design", ISBN 0-13-463894-8, 1995
T.L.Floyd: "Digital Fundamentals", Prentice Hall, ISBN 0-13-080850-4, 2000

Úvod

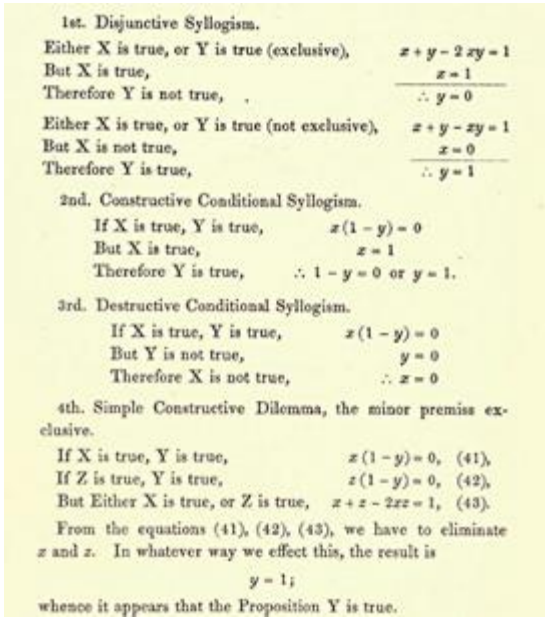
Verze: 20200202

Administrativa

- Přednášející, cvičící
  - Otto Fučík, fucik@fit.vutbr.cz
  - Jan Kořenek, korenek@fit.vutbr.cz
  - Tomáš Martínek. martinto@fit.vutbr.cz
- Kontrola znalostí
  - Půlsemestrální zkouška (25 bodů)
  - Projekt (20 bodů) – zápočet
  - Semestrální zkouška (55 bodů)
- Pravidla
  - Min. 5 bodů z projektu - podmínka nutná pro získání zápočtu
  - 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, $\therefore x = 0$

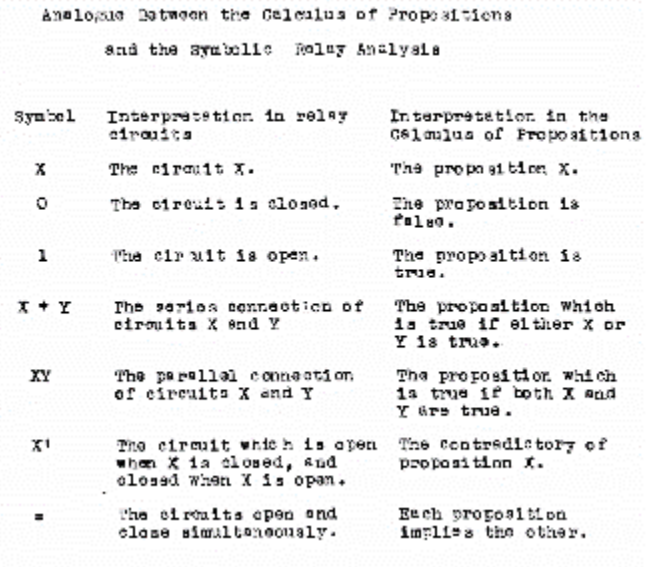
4th. Simple Constructive Dilemma, the minor premiss exclusive.

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.

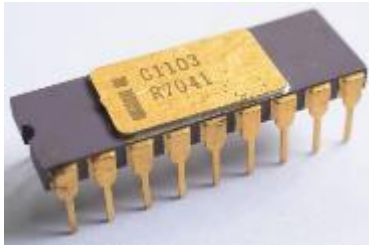


Analogue Between the Calculus of Propositions and the Symbolic Relay Analysis

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? (biotechnologie, nanotechnologie, kvantové jevy…)



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.

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

Aplikačně specifické a univerzální výpoč. systémy

- Aplikačně specifické výpočetní systémy
  - Funkce je optimalizována pro danou (specifickou) funkci
  - Vysoká efektivita výpočtů (velká výkonnost, nízké energetické nároky)
  - Složitý návrh a výroba = vysoká cena (vyplatí se při hromadné výrobě)
  - Lze modifikovat (programovat, konfigurovat) jen v omezené míře
- Univerzální výpočetní systémy (počítače)
  - Určeny pro obecné použití (univerzální)
    - Amortizace návrhu a výroby výpočetního stroje
    - Mohou být programovány = levný návrh aplikací (software)
  - Nižší efektivita výpočtů oproti aplikačně specifických systémům
  - Dnes jsou často vestavěny do větších systémů (anglicky Embedded Systems), kde vykonávají specifickou funkci

Tvorba složitých systémů

- Složitý systém
  - Hierarchicky uspořádaná struktura, ve které jednotlivé subsystémy (komponenty) komunikují informace přes rozhraní (přenáší informaci) pomocí komunikačního média (vodiče)
  - Komponenty a komunikace mezi nimi - čím jednodušší, tím menší možnost chyb (1 bit – nejmenší hodnota)
- Číslicové systémy
  - Jsou sestaveny z komponent, které pracují a komunikují pouze s 1bitově reprezentovanou informací (nejjednodušší možná forma = jednoduché, spolehlivé, levné…)
  - Umožňují tvorbu výpočetních strojů (počítače), které zpracovávají binárně kódované informace pomocí logických obvodů (Booleova algebra)

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

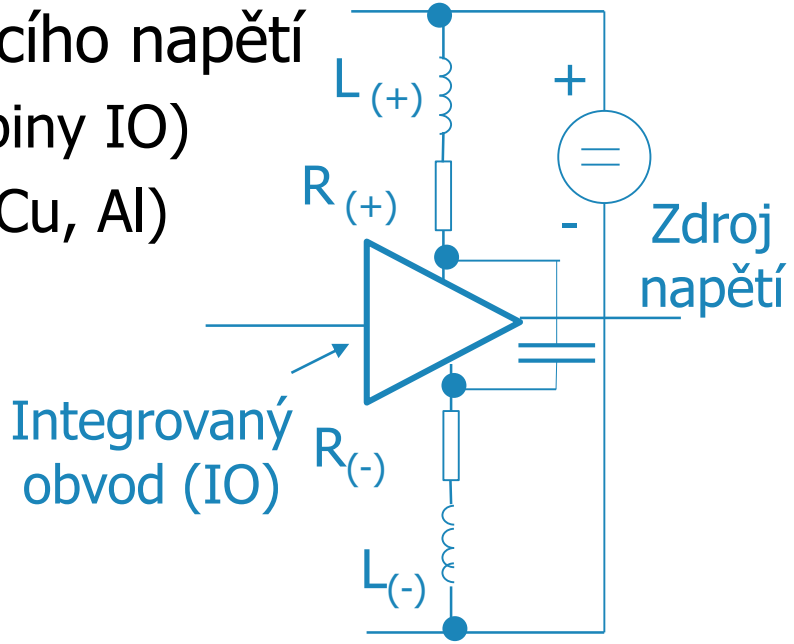
- Informace (logické úrovně) se v číslicovém systému přenáší vodiči
  - Ve vodiči se může indukovat rušivé napětí, které se přičítá k platné logické hodnotě => hodnota napětí se může dostat do zakázané oblasti a obvod pak nepracuje tak, jak je očekáváno
- Číslicové obvody proto musí být navrženy tak, aby odolávaly rušení z různých vnějších i vnitřních (viz dále) zdrojů
  - Hovoříme o tzv. elektromagnetické kompatibilitě – jedna ze základních vlastností všech elektronických zařízení
- Každé elektronické zařízení musí být (dokonce ze zákona) certifikováno 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

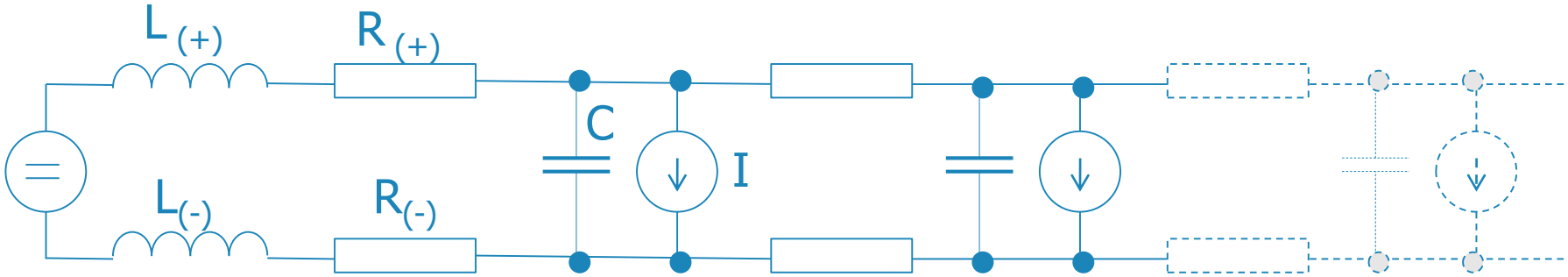
Rušení: Odrazy na vedení, odběr proudu

- 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

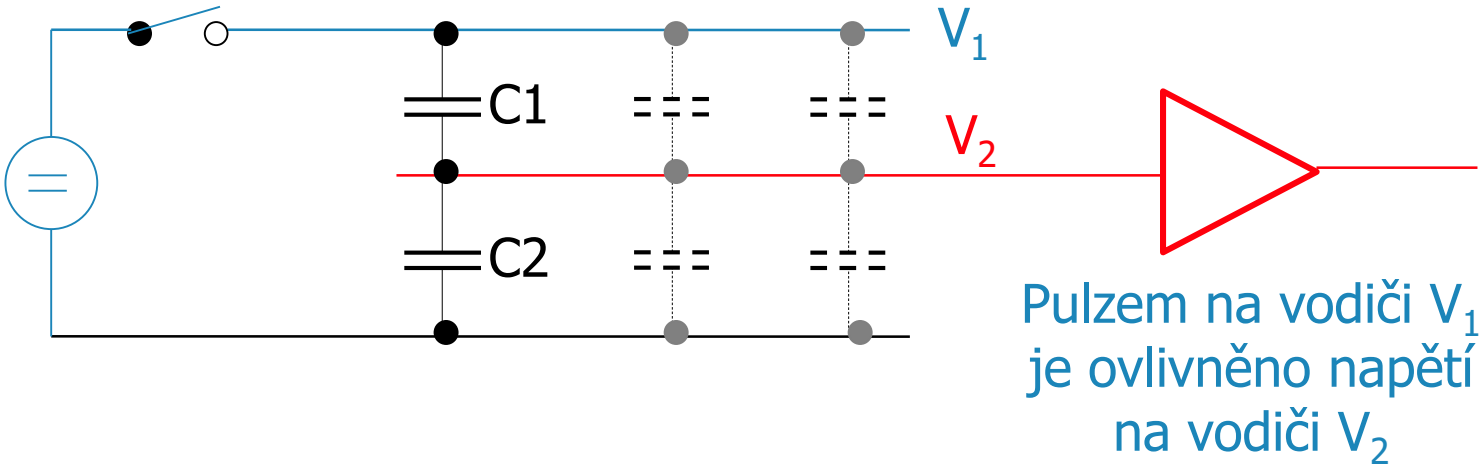
- Náhradní schéma rozvodů napájecího napětí
  - L – indukčnost (přívodní vodiče, piny IO)
  - R – odpor propojovacích vodičů (Cu, Al)
  - C – parazitní kapacity mezi vodiči
  - I – odebíraný proud obvodu
- Zdroje rušivého napětí
  - $U_R = I \cdot R$, $U_L = di/dt$
  - LRC tvoří rezonanční obvod
  -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





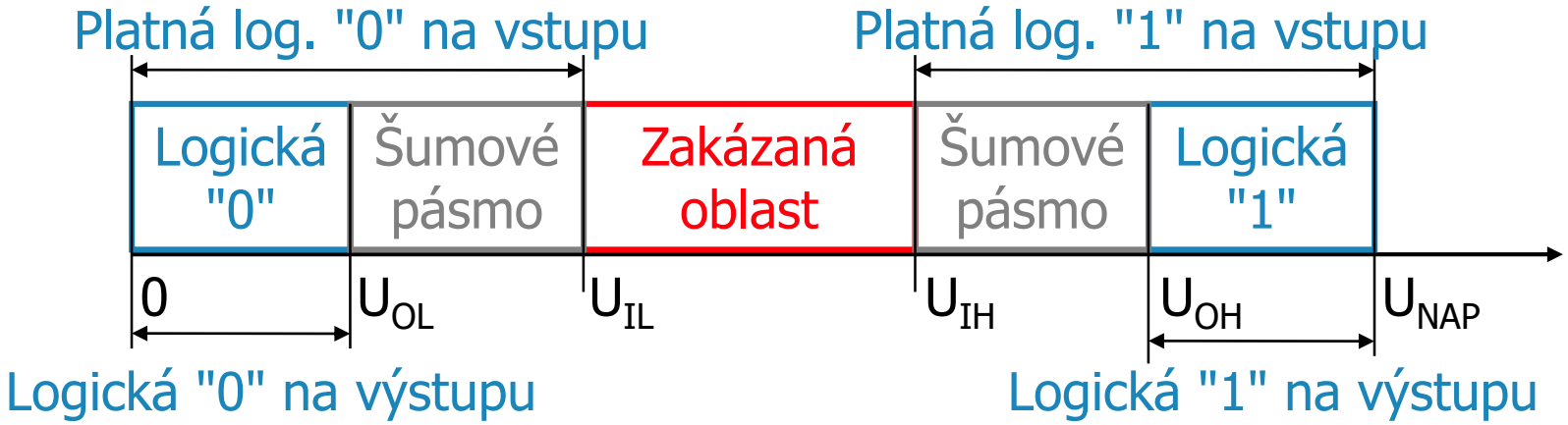
Rušení: Přeslechy

- 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



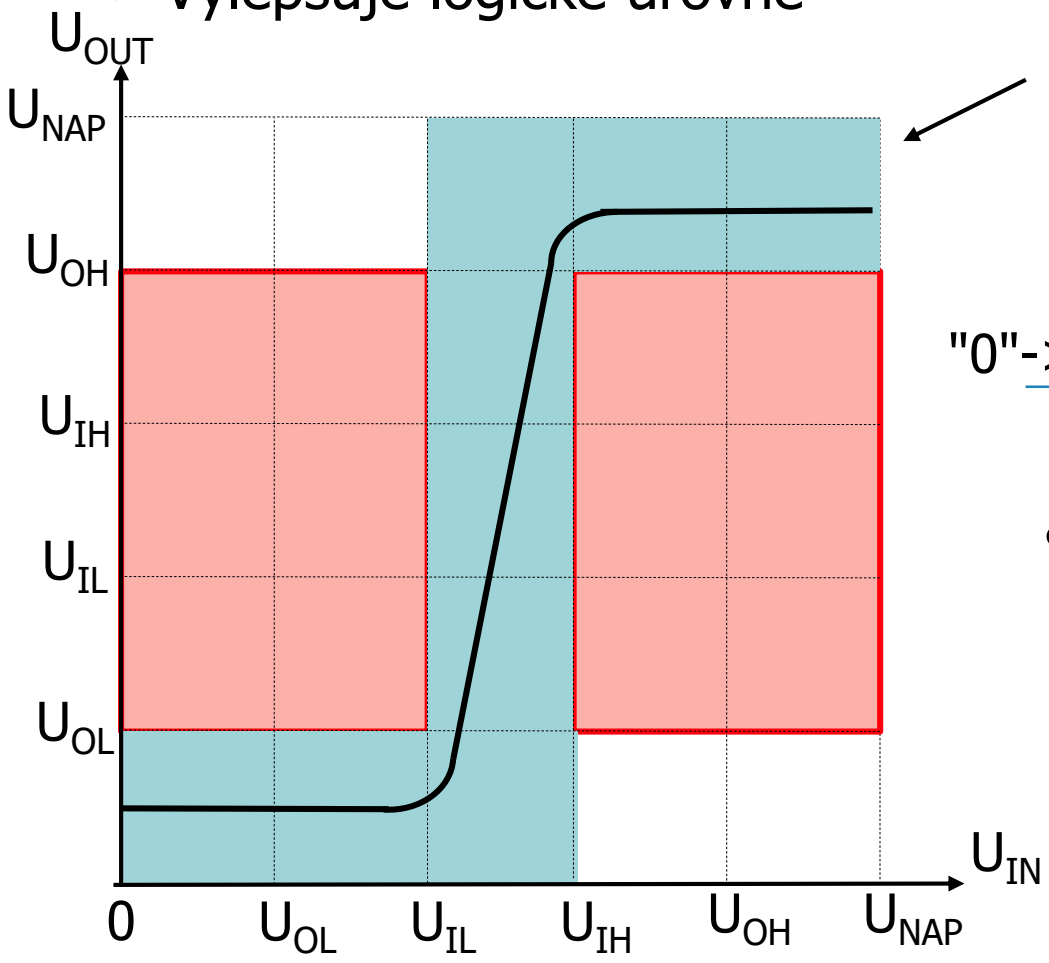
Rušení: Omezení vlivu

- Šumové pásmo
  - Obvody musí "vylepšovat" hodnoty napěťových úrovní mezi vstupem a výstupem
- Obvody díky tomu akceptují i jistou úroveň rušení
  - Logická "0": napětí v rozsahu 0 až U_{IL}, pásmo akceptovaného šumu U_{IL}-U_{OL}
  - Logická "1": napětí v rozsahu U_{IH} až U_{NAP} (napájecí napětí), pásmo akceptovaného šumu U_{OH}-U_{IH}
- Obvod generuje "vylepšené" logické úrovně



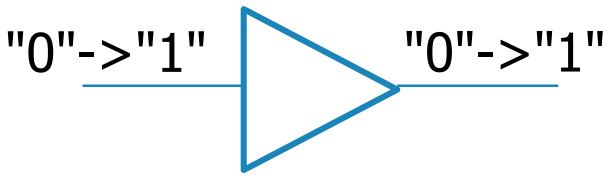
Obvod odolný proti rušení: Příklad

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



Převodní charakteristika

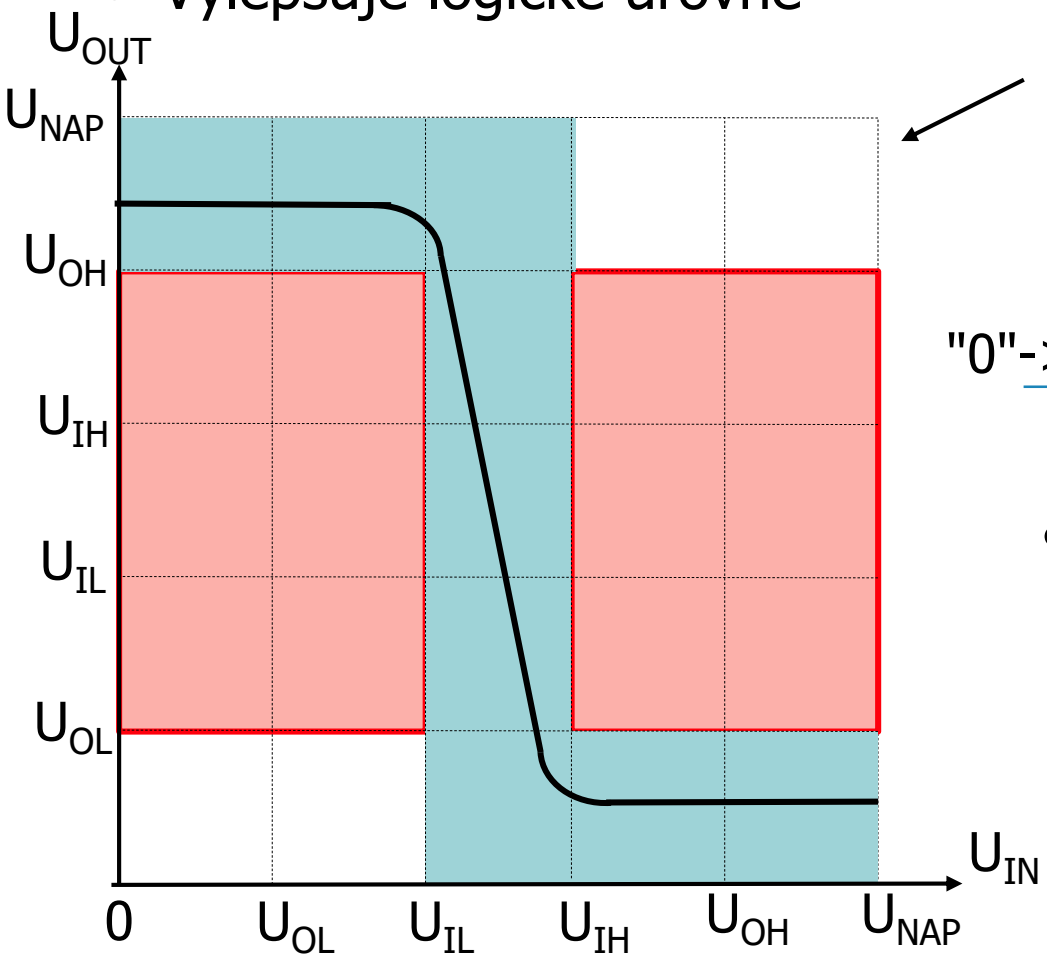
Symbol



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

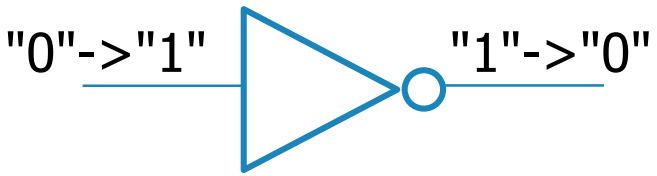
Obvod odolný proti rušení: Příklad

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



Převodní charakteristika

Symbol



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

Číslicové systémy

- Limity
  - Dnešní tranzistory jsou schopny pracovat velmi rychle (stále se zmenšující rozměry díky rostoucí hustotě integrace, viz Mooreův zákon)
  - Vodiče mezi tranzistory nelze eliminovat, neboť musí propojit různé části obvodu => zpoždění, se kterým je třeba počítat
- Analogové chování
  - Součástky (tranzistory) jsou používány ve spínacím režimu (on-off), což umožňuje při návrhu abstrahovat od jejich analogové podstaty
  - Při provozu však musíme brát ohled i na jejich analogové chování (vliv omezujeme vhodnou konstrukcí logických členů, viz dále) – k přechodům mezi stavy "on" a "off" nedochází okamžitě

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 IO
  - Umíme je efektivně využít?

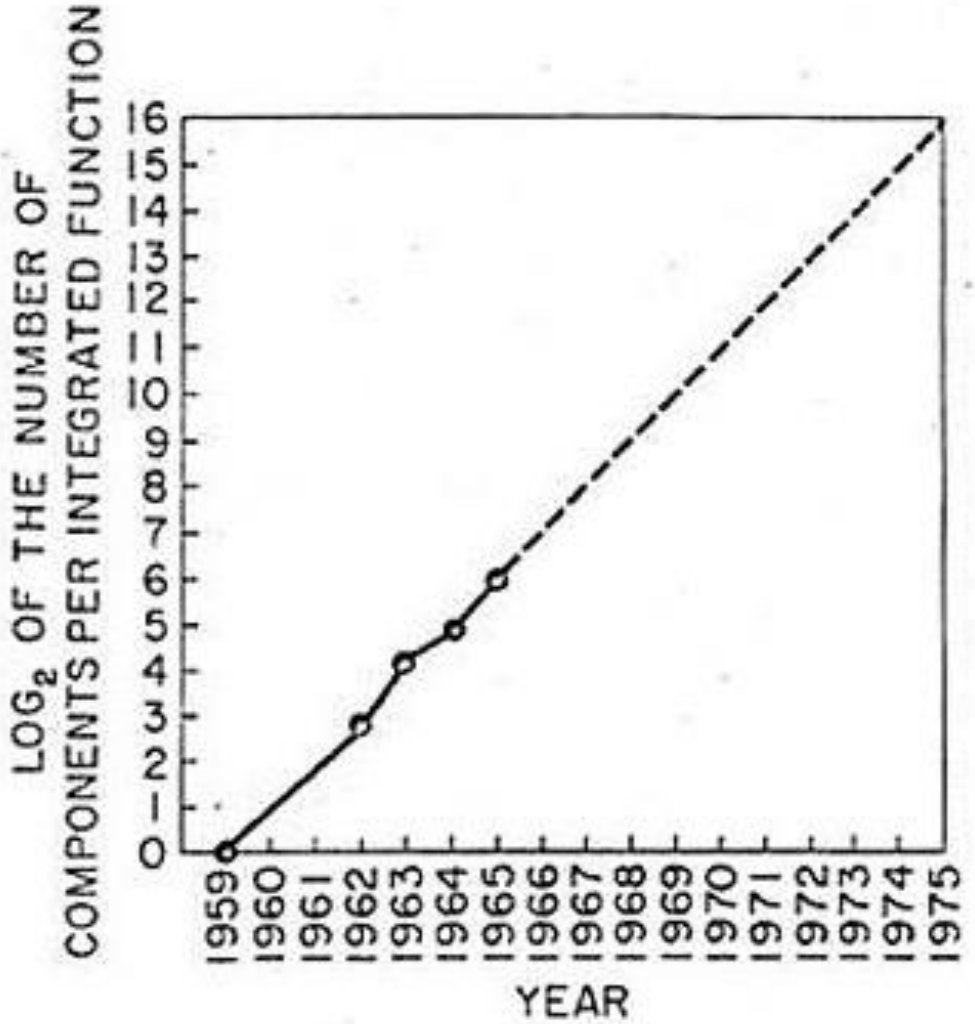
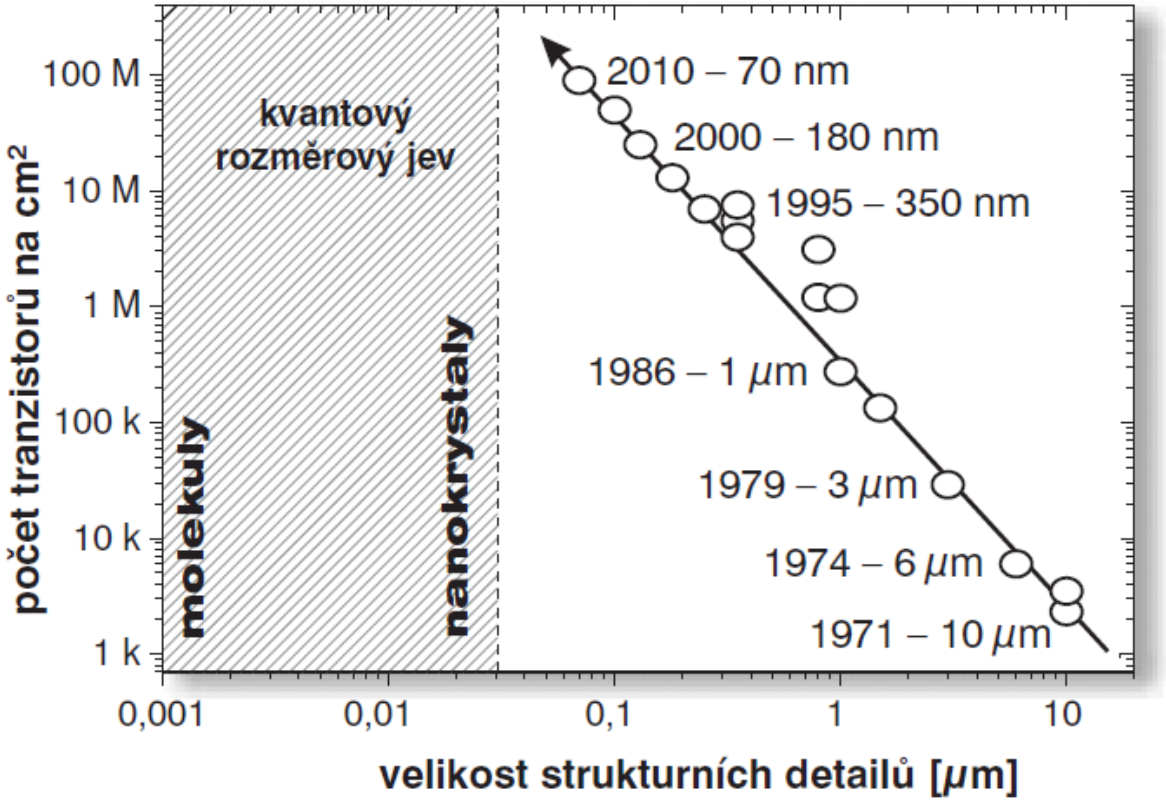


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

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

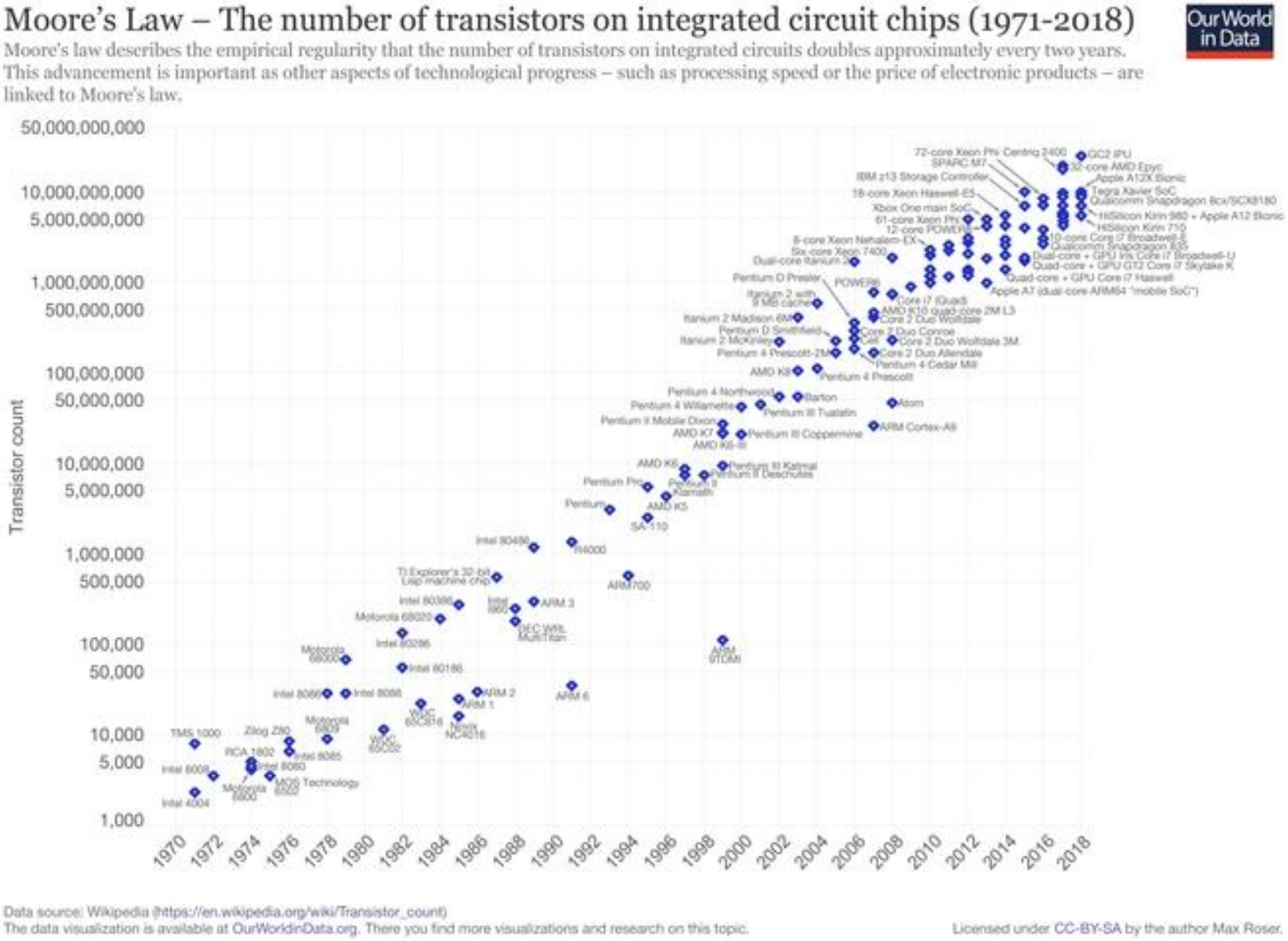
Mooreův zákon: Predikce z roku 2001

- Hustota tranzistorů na čipech jako funkce velikosti minimální strukturní jednotky obvodu (osy jsou log.)
- 1960 - MOSFET laboratorní tranzistor 20 μm (délka hradla)
- 1969 - MOS SRAM komerční paměť 12 μm

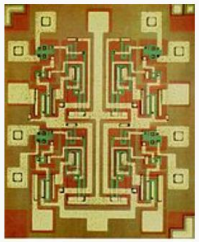


[Zdroj: Valenta, J.: Integrovaný obvod - základní kámen informační revoluce, Vesmír, leden 2001, p. 24-29.]

Mooreův zákon: Stav v roce 2018



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

Future

3 nm – *~2021*
2 nm – *~2024*

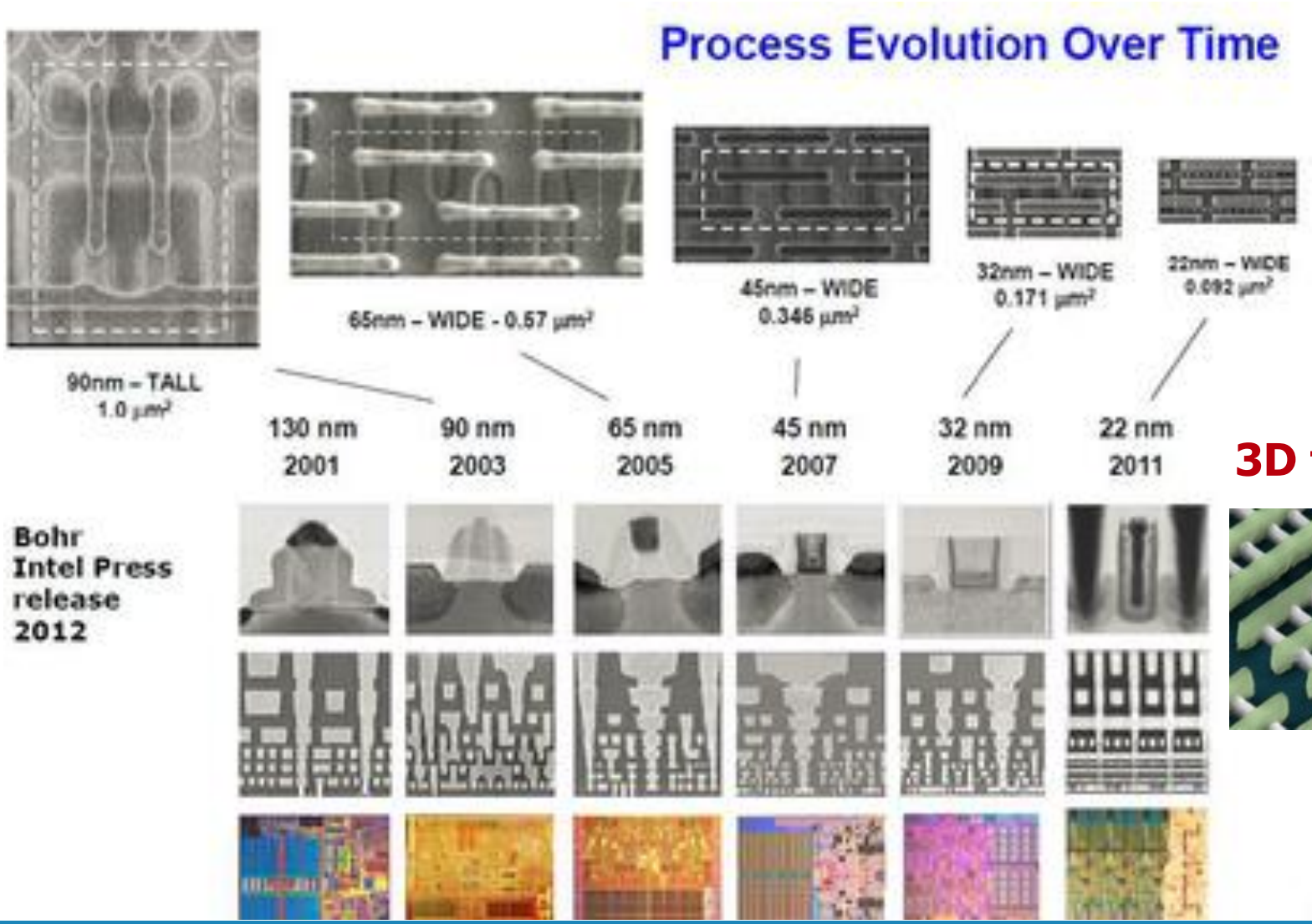
Mooreův zákon: Predikce 2021

	Samsung	TSMC	IRDS roadmap 2017	
Process name (nm)	5LPE	N5	7	5
Transistor density (MTr/mm^2)	127	173	222 (37×6)	300 (50×6)
SRAM bit-cell size (μm^2)	0.026	0.017–0.019	0.027	0.020
Transistor gate pitch (nm)	57	48	48	42
Interconnect pitch (nm)	36	30	28	24
Production year	2018	2019	2019	2021
Based on a 6T SRAM 111 cell				

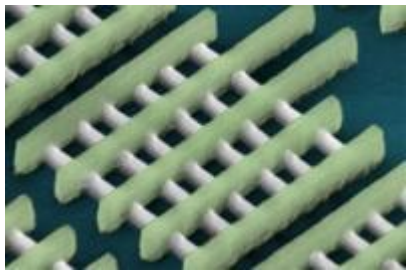
Zdroj: https://en.wikipedia.org/wiki/5_nm_process

Mooreův zákon: Ilustrace zmenšování rozměrů

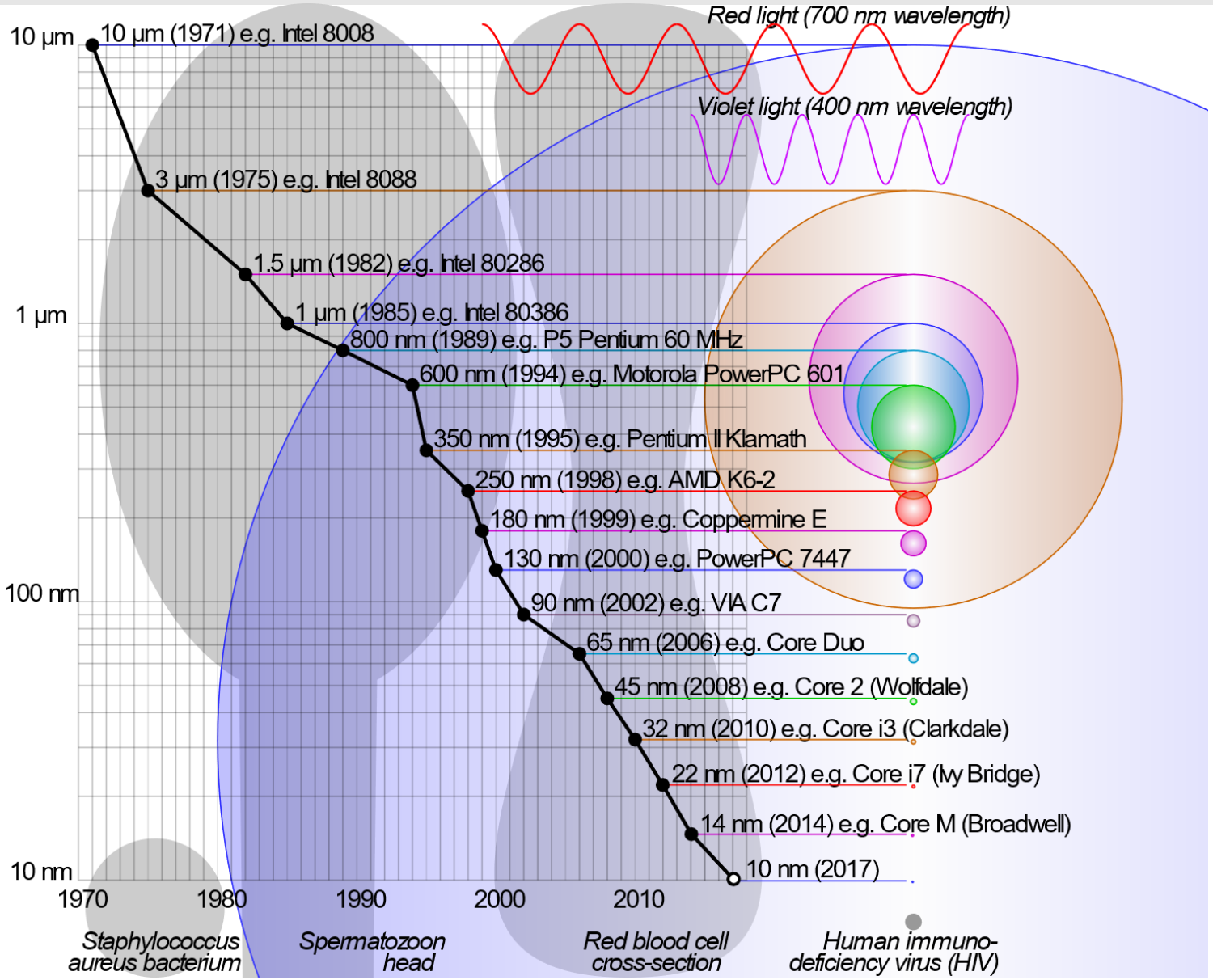
- Příklad: Intel



2011
3D tranzistory

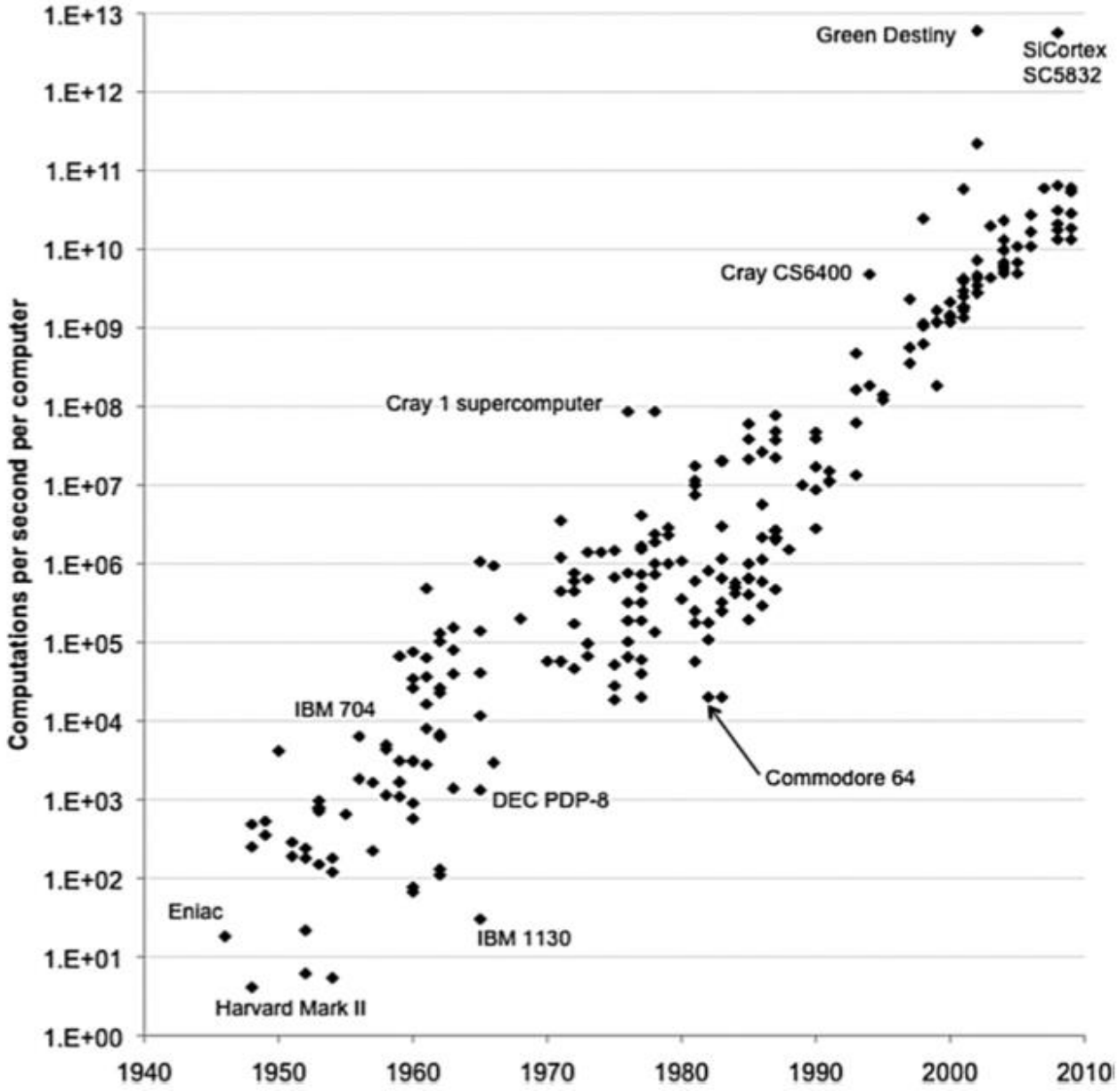


Mooreův zákon: Ilustrace zmenšování rozměrů



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

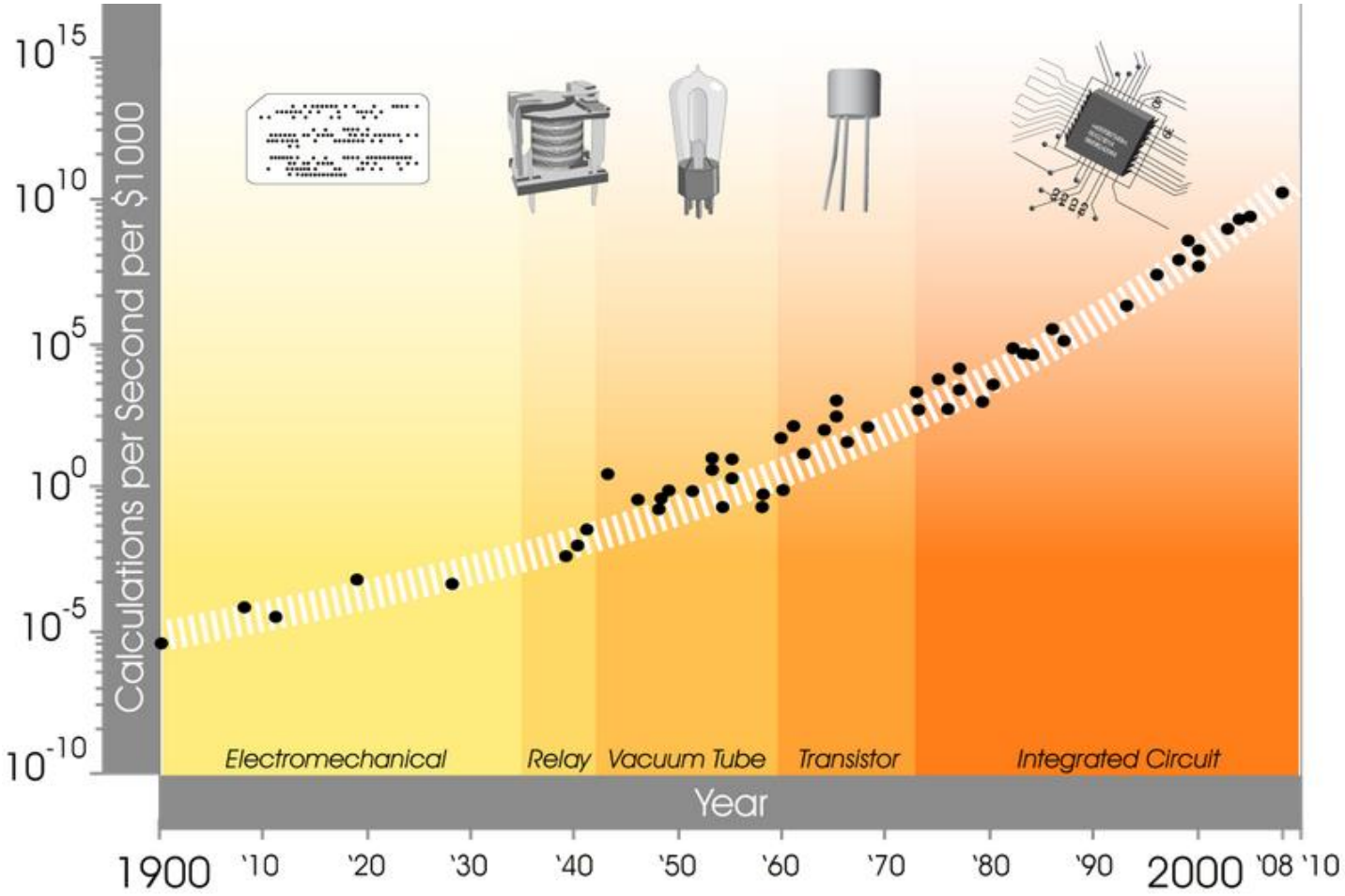
Počet výpočtů za sekundu



[Zdroj: Koomey, Berard, Sanchez, and Wong (2011) – Implications of Historical Trends in the Electrical Efficiency of Computing. In IEEE Annals of the History of Computing, 33, 3, 46–54.]

Počet výpočtů za sekundu (za $1000)

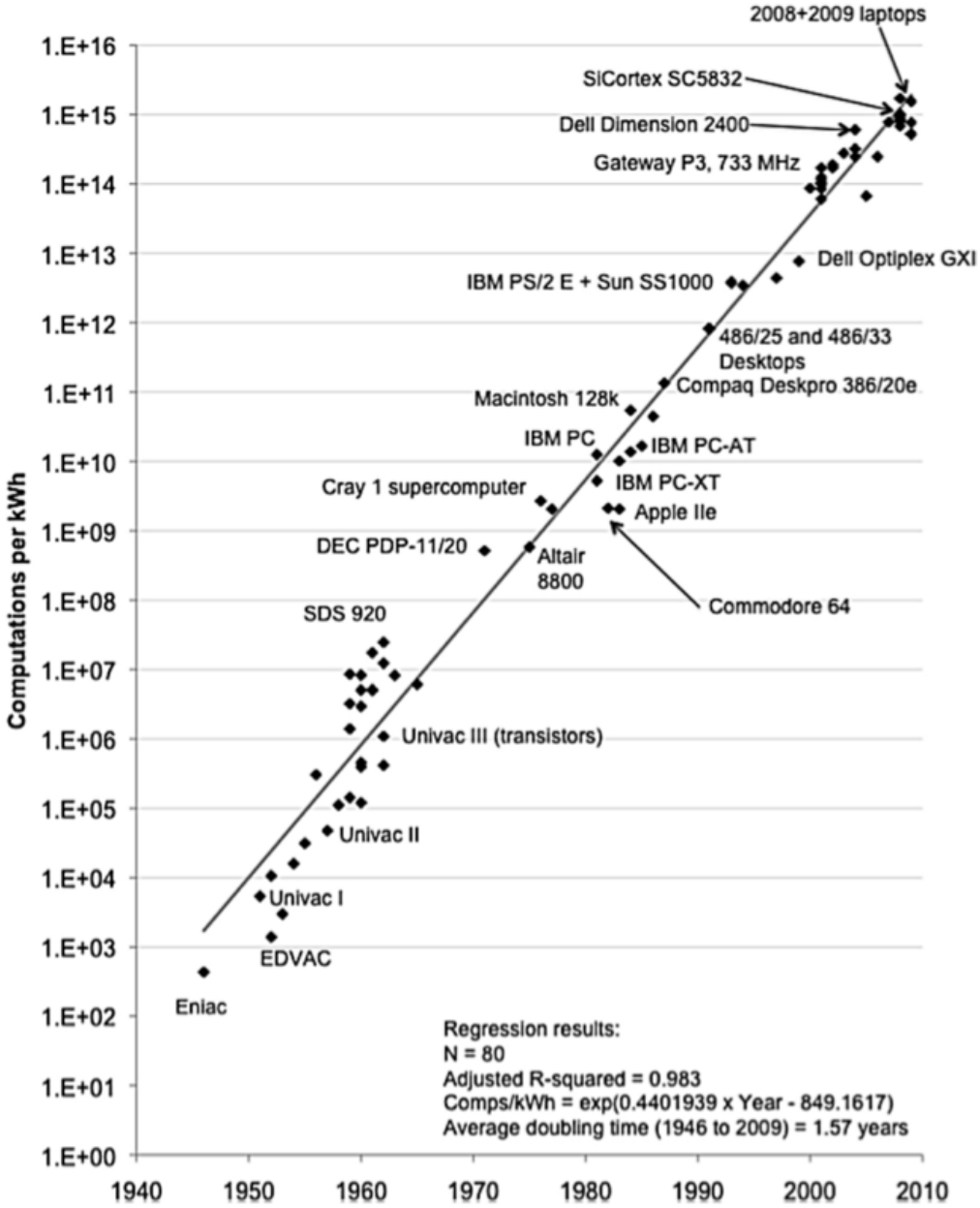
- Např. v roce 2013 měl běžný laptop srovnatelný výkon s nejvýkonnějším počítači poloviny 90tých let



[Zdroj: http://www.kurzweilai.net/exponential-growth-of-computing#!prettyPhoto]

Energetická náročnost výpočtů

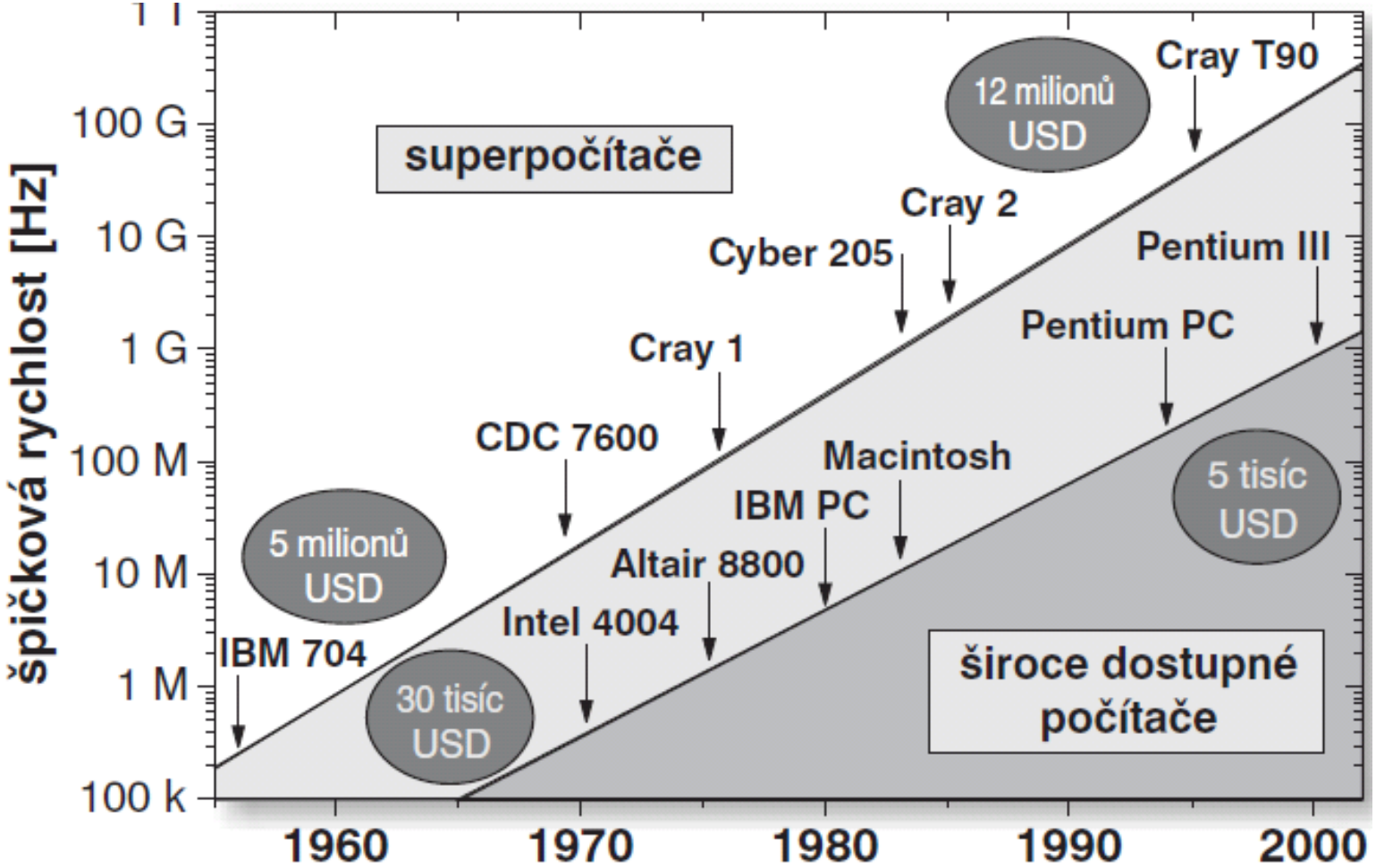
- Počet výpočtů na kWh
- Měřeno při maximální výpočetní výkonnosti
- Mezi lety 1946 až 2009 se energetická efektivita výpočtů zdvojnásobuje cca každého 1,6 roku



[Zdroj: Koomey, Berard, Sanchez, and Wong (2011) – Implications of Historical Trends in the Electrical Efficiency of Computing. In IEEE Annals of the History of Computing, 33, 3, 46–54.]

Cena počítačů je téměř konstantní, výkon ale roste

- Cena posledních modelů počítačů je téměř konstantní, i když jejich výkonnost roste (viz Mooreův zákon)

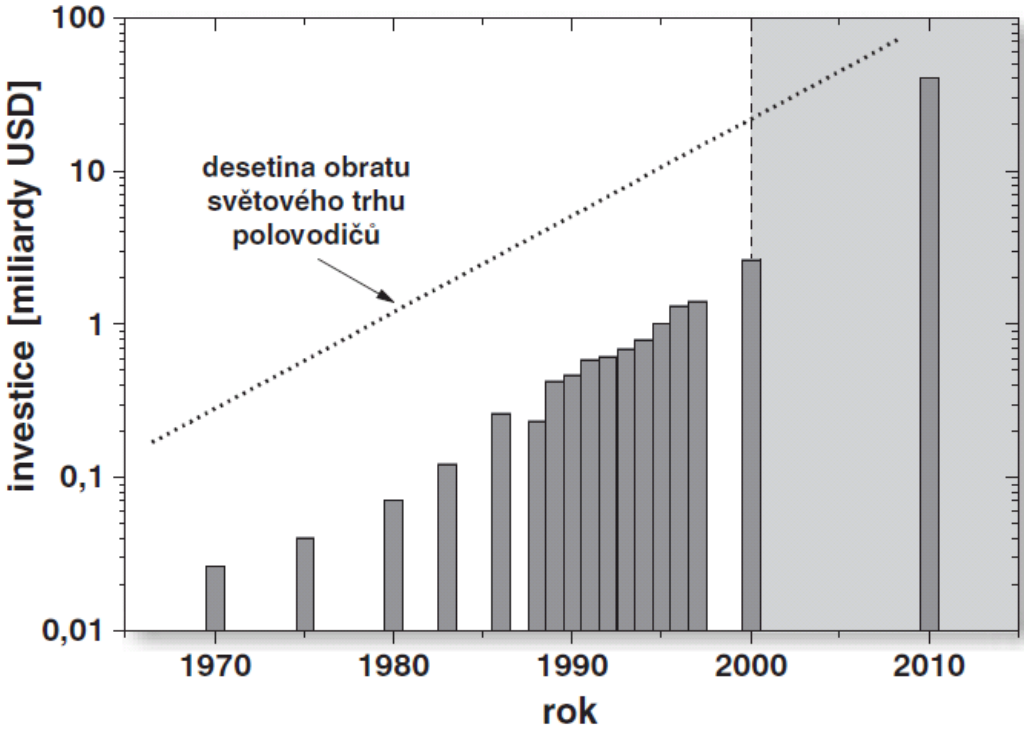


[Zdroj: Valenta, J.: Integrovaný obvod - základní kámen informační revoluce, Vesmír, leden 2001, p. 24-29.]

Rockův zákon: Investice do výroby čipů

- Tzv. 2. Mooreův zákon
  - Exponenciální růst investic (dnes několik miliard USD) nutných pro zavedení nové technologie výroby integrovaných obvodů
  - Investice do nových zařízení na výrobu čipů se zdvojnásobuje každé 4 roky
  - 2015 - 14 miliard USD

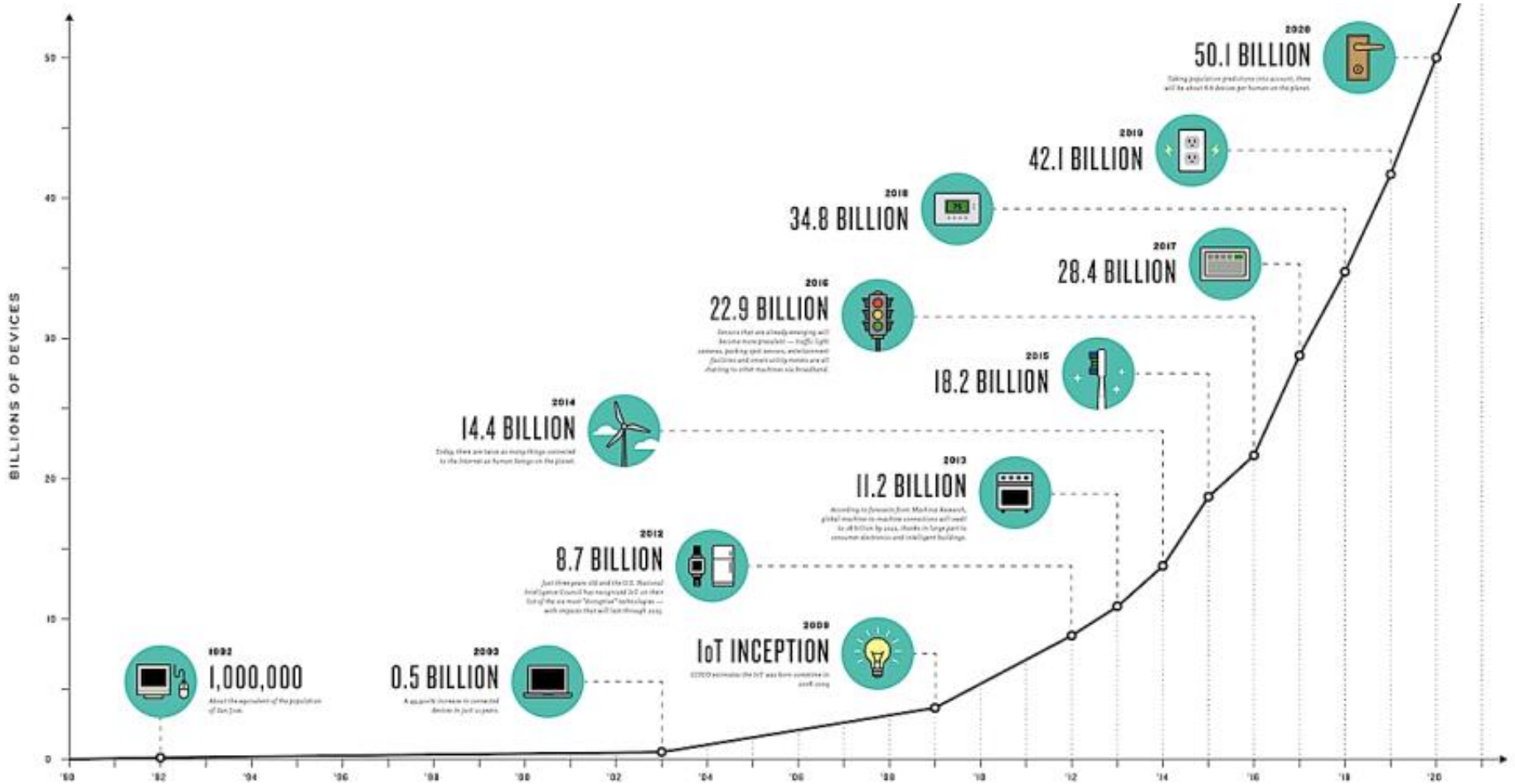
- Hodnota investic se blíží jedné desetině celkového světového obratu polovodičového průmyslu



[Zdroj: Valenta, J.: Integrovaný obvod - základní kámen informační revoluce, Vesmír, leden 2001, p. 24-29.]

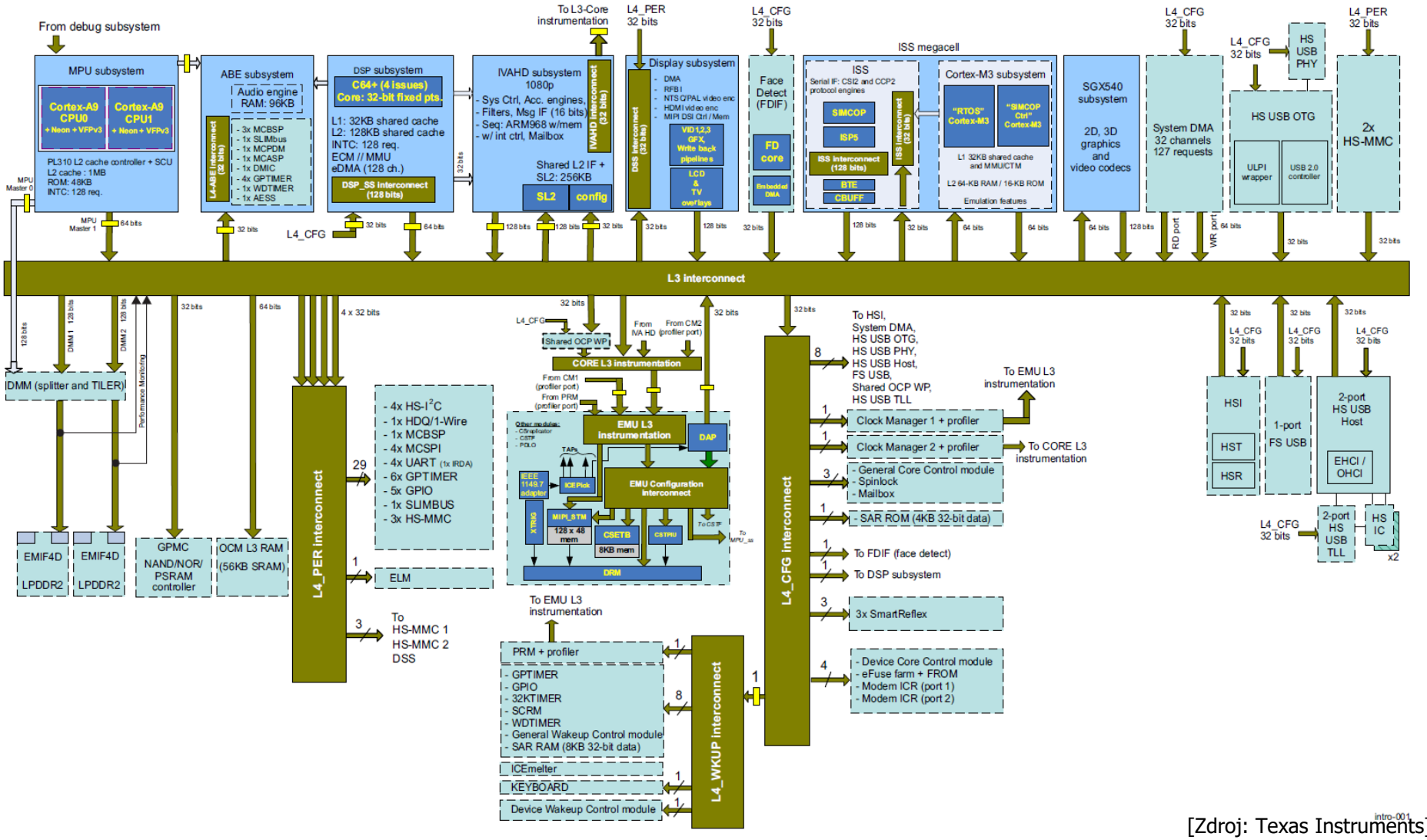
Nejen složitost komponent roste, ale i jejich počet

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



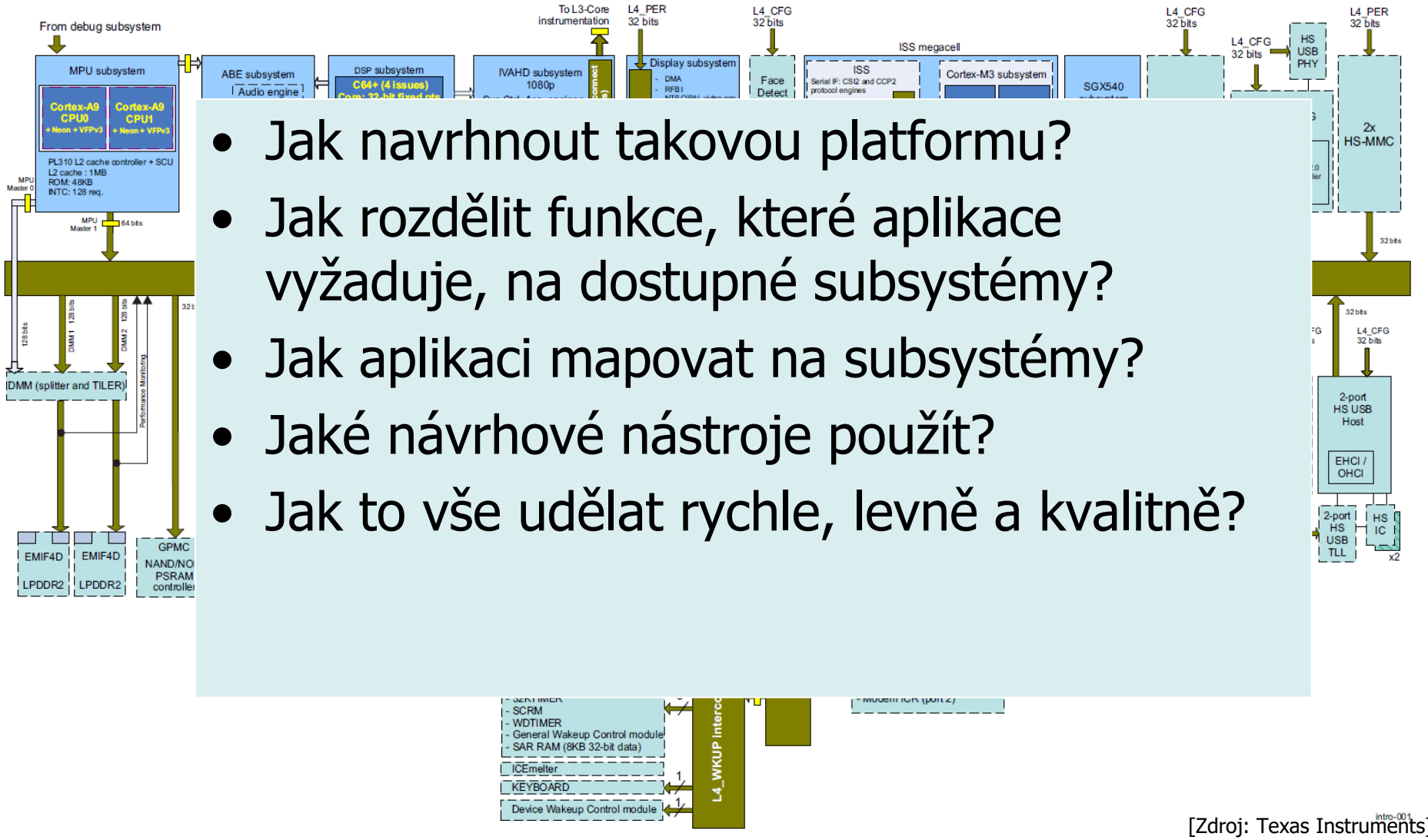
[Zdroj: J. Straw: http://idisrupted.com/disrupted-electronics-internet-things-may-create-moores-law-steroids/]

- Platforma OMAP4430 - SoC pro mobilní zařízení



[Zdroj: Texas Instruments]

- Platforma OMAP4430 - SoC pro mobilní zařízení

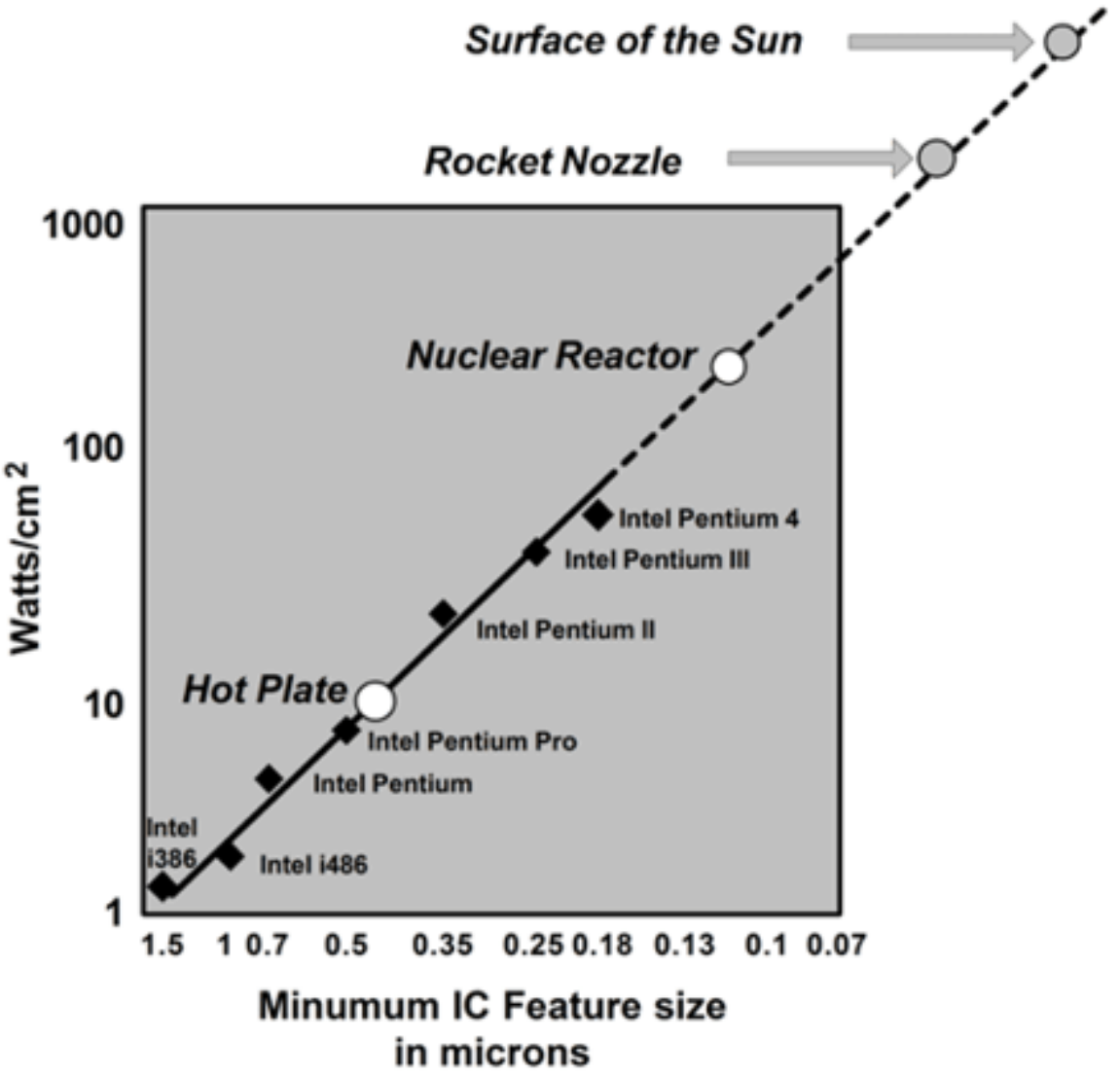


- Jak navrhnout takovou platformu?
- Jak rozdělit funkce, které aplikace vyžaduje, na dostupné subsystémy?
- Jak aplikaci mapovat na subsystémy?
- Jaké návrhové nástroje použít?
- Jak to vše udělat rychle, levně a kvalitně?

[Zdroj: Texas Instruments]

Limity: Využitelná výkonnost a odvod tepla

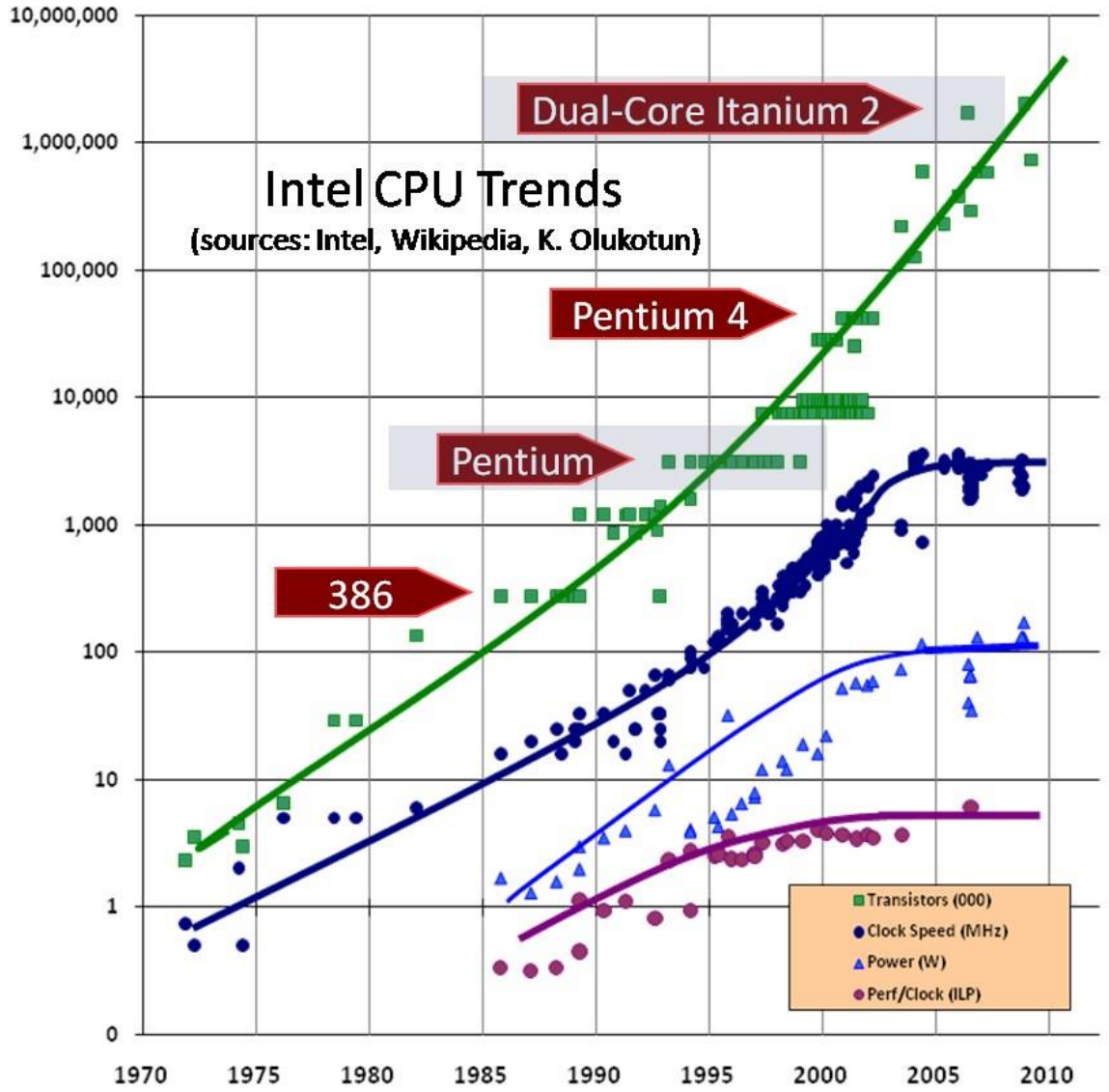
- Efektivní (užitečná) výkonnost
  - Roste přibližně s druhou odmocninou jejich složitosti (Pollackovo pravidlo)
- Hustota energie (Power Density) roste s výkonností
  - Tepelné ztráty limitují výpočetní výkonnost (odvod tepla)



[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.

Skončí platnost Mooreova zákona?

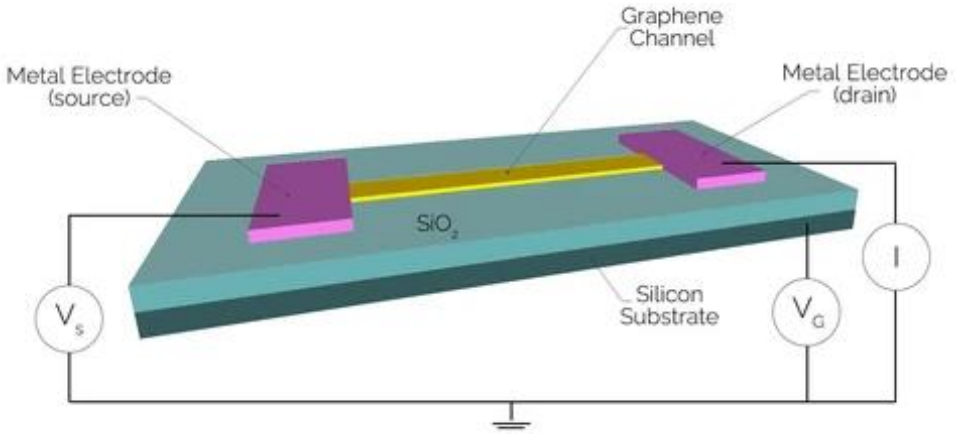
- Intel's former chief architect Bob Colwell:
  - „Moores Law will be dead and buried by 2020 or 2022. The Fabrication process will either be 7nm or 5nm and its killer will not be Physics but rather Economics."



[Zdroj: U. Pirzada, „Moores Law will be Dead by 2020 Claim Experts – Last Fab Process for CPU/GPUs to be 5nm", https://wccftech.com/moores-law-will-be-dead-2020-claim-experts-fab-process-cpug-pus-7-nm-5-nm/

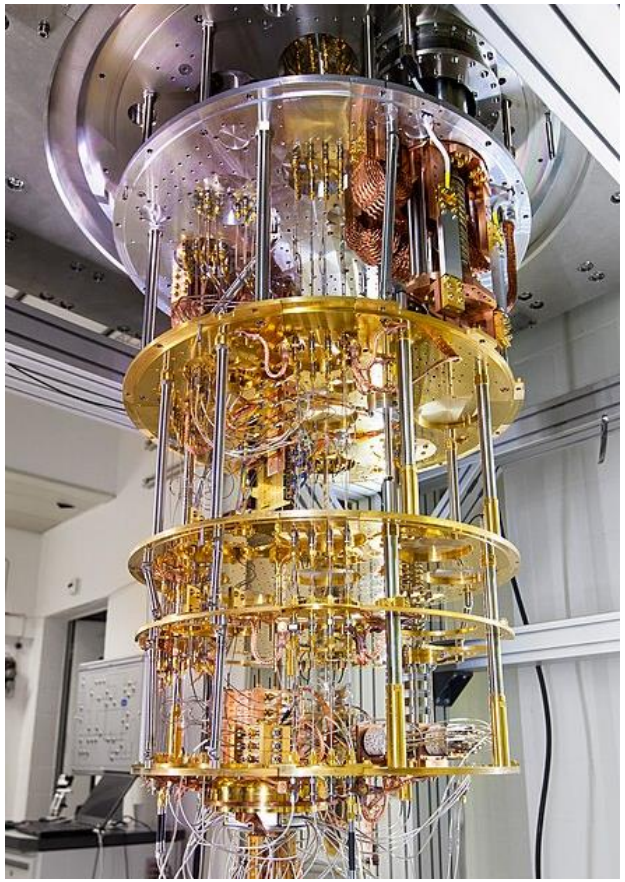
Nejspíš ano (podhodnocením dnešních možností)

- Graphene Field Effect Transistor – GFET (2004)
  - 475 GHz (2013)



[Zdroj: M. Bolza, „What Are Graphene Field Effect Transistors (GFETs)?„, https://www.graphenea.com/pages/what-are-graphene-field-effect-transistors-gfets

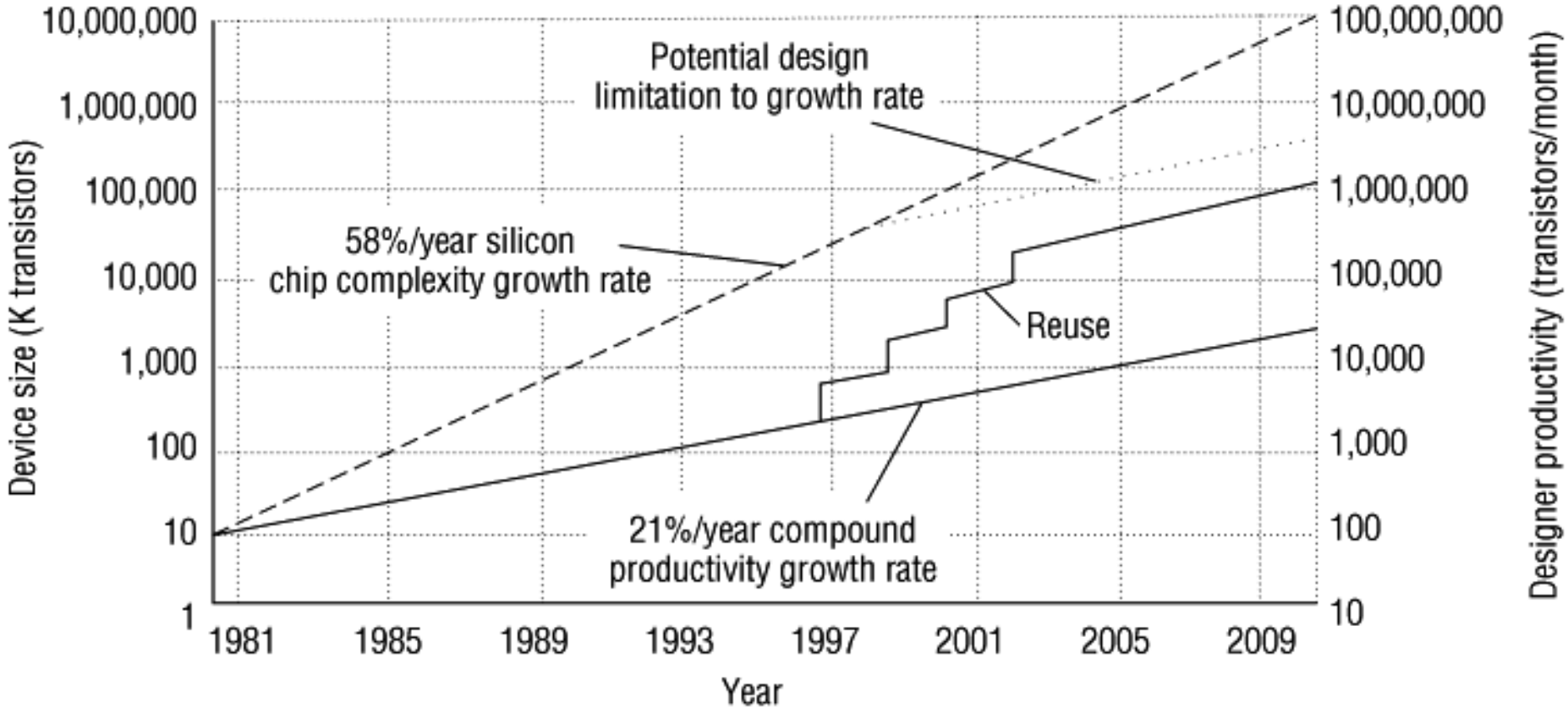
- Quantum Computing (1980)
  - IBM quantum computer (chlazeno pod 1 Kelvin)



[Zdroj: https://en.wikipedia.org/wiki/Quantum_computing

Limity: Produktivita práce při návrhu

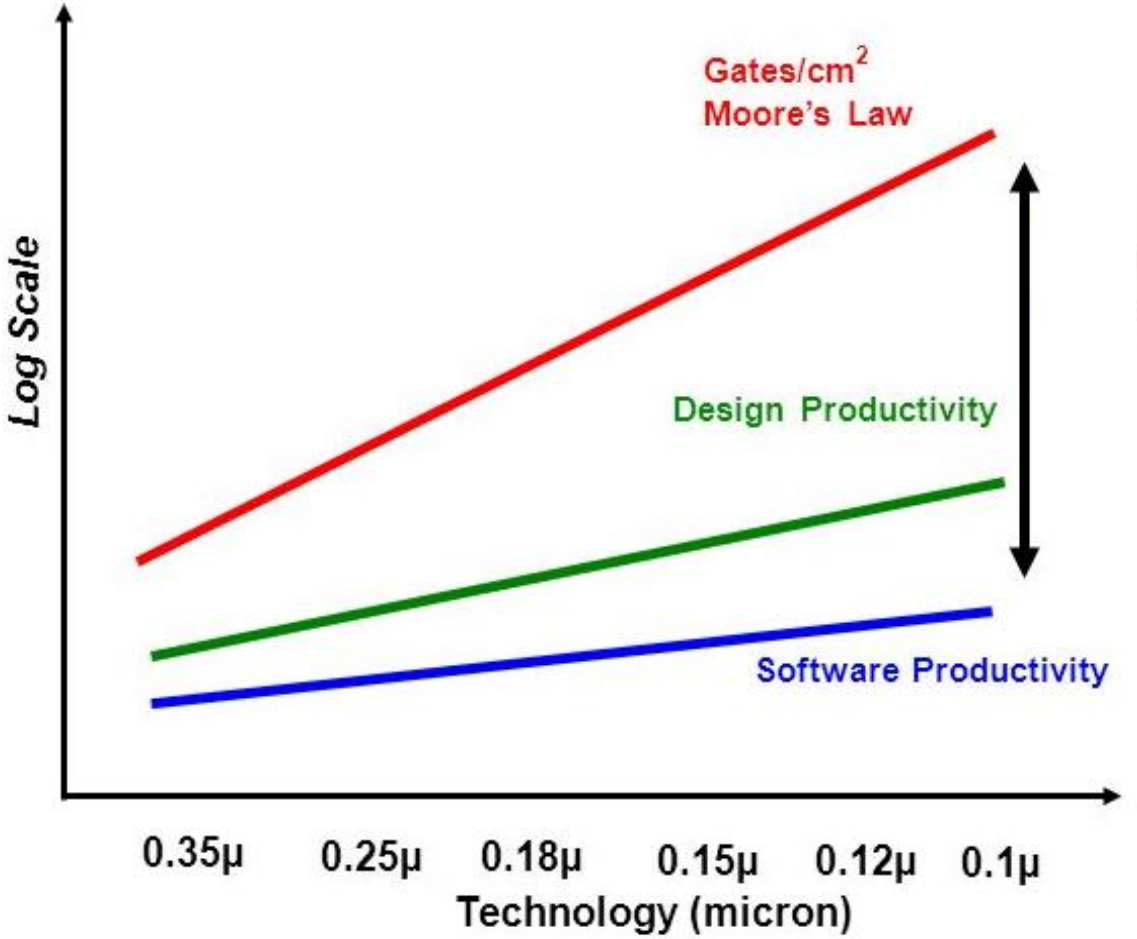
- Složitost integrovaných obvodů roste o cca 60 % ročně
- Produktivita práce pouze o cca 20 % ročně



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

Limity: Produktivita práce při návrhu HW a SW

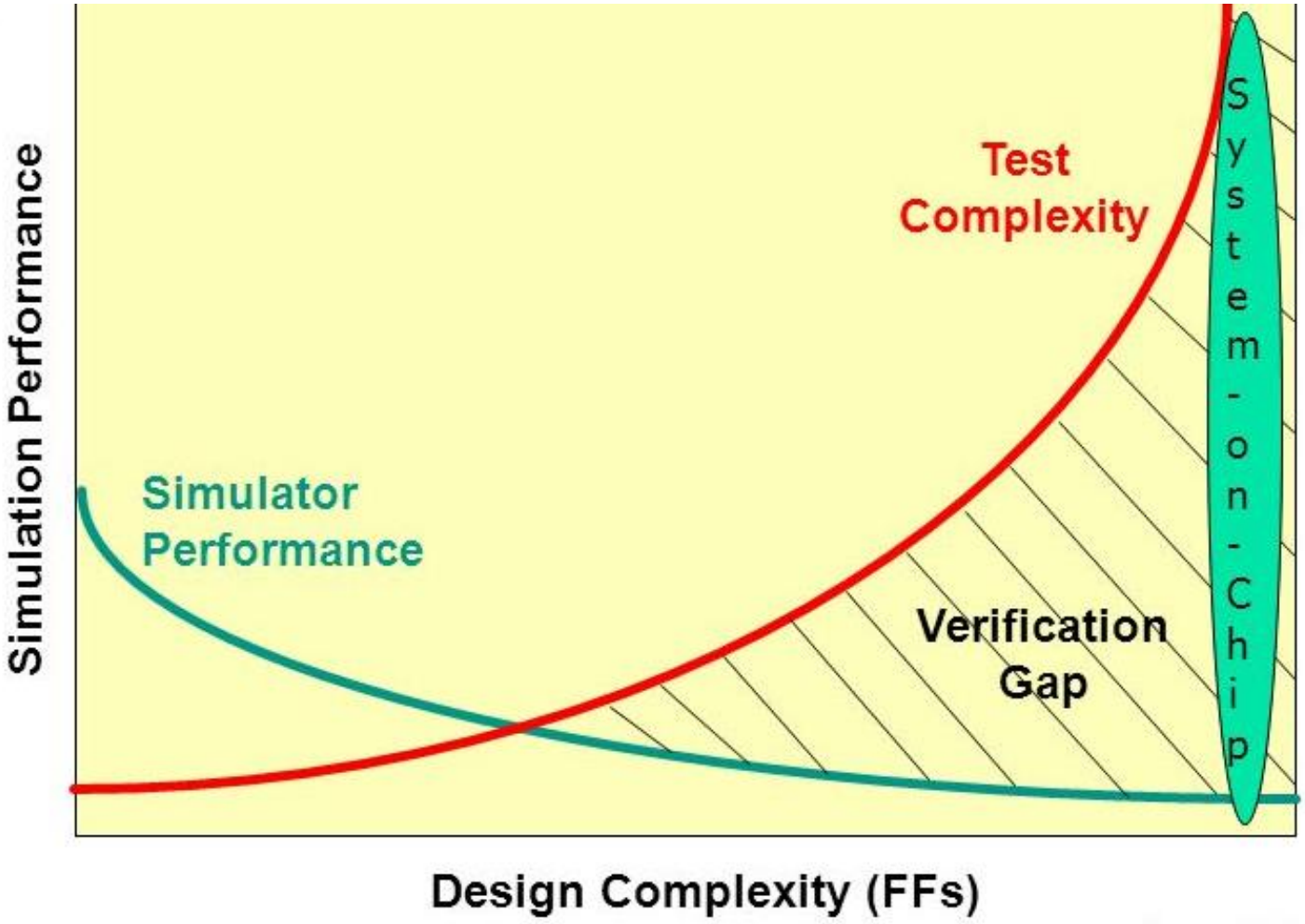
- S rostoucí složitostí obvodů klesají naše schopnosti jejich efektivního využití



[Zdroj: Peter Faber: Embedded Electronics Large to small… Systems, 2015.]

Limity: Složitost návrhu a možnosti jeho testování

- S rostoucí složitostí obvodů klesají naše schopnosti jejich efektivního testování



[Zdroj: Cadence]

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, sociální, etické, zdravotní, bezpečnostní, výrobní, servisní a další aspekty

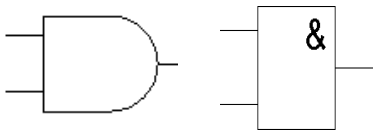
Logické obvody

- 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
  - Se skládají 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)

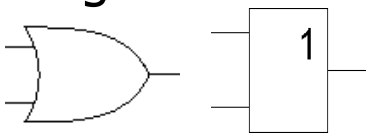
Logické členy

- 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 resp. NOR = OR s invertorem na výstupu)
- Značení
  - Čtvercové značky - funkce logického členu je označena znaky "&" je pro funkci AND, "1" pro funkci OR
  - Značky složené z křivek - rozšířené ve většině profesionálních systémech pro návrh logických obvodů (každý způsob značení má své výhody a nevýhody)

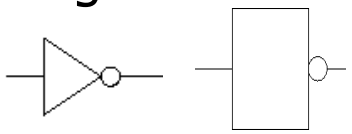
Log. člen AND



Log. člen OR

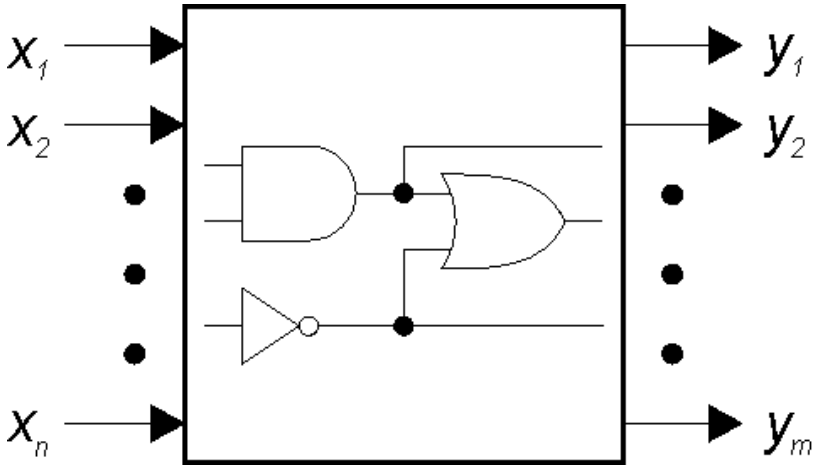


Log. člen NOT



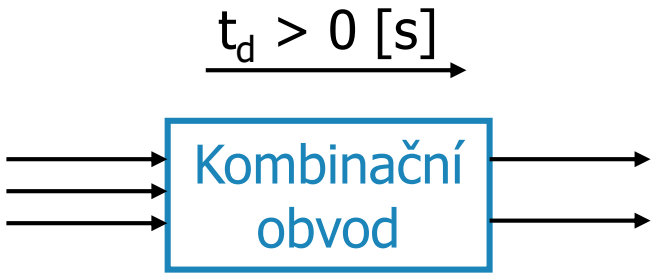
Kombinační logické obvody

- Hierarchicky uspořádaná struktura, ve které jednotlivé komponenty zpracovávají a mezi sebou komunikují informace 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



Kombinační logické obvody

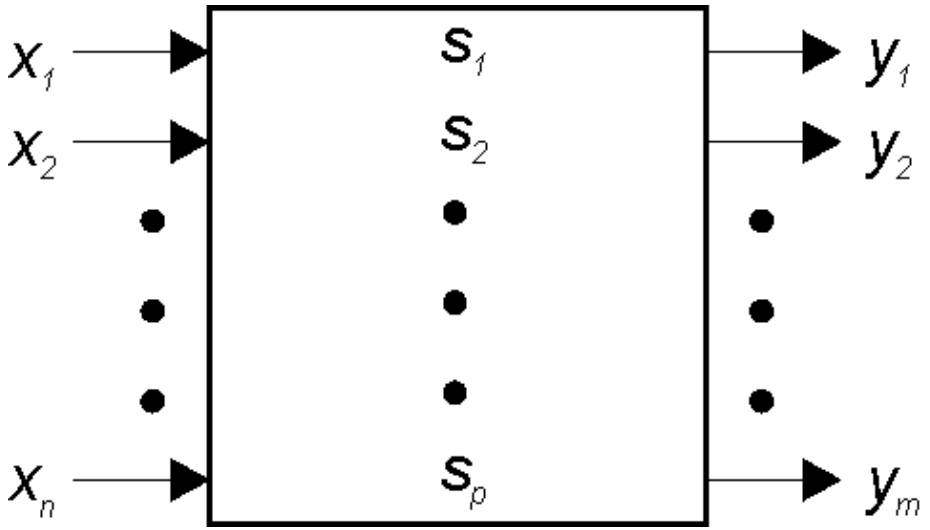
- Specifikace chování (behaviorální, funkce)
  - Definice hodnoty (logické úrovně) každého výstupu pro všechny kombinace vstupních hodnot (logických úrovní)



- Specifikace časování
  - Každý fyzicky realizovaný obvod má zpoždění t_d
    - Definováno např. jako nejhorší potřebná doba pro výpočet (vygenerování) platných logických hodnot výstupů od okamžiku, kdy na vstupech budou platné a stabilní logické hodnoty
  - Při návrhu se zpoždění často zanedbává ($t_d = 0$)
    - Prakticky představuje zpoždění jeden z největších problémů při implementaci (rychlost výpočtů, hazardy – viz dále, vliv prostředí, atd.)

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 být různá
- Aktuální hodnoty stavových proměnných
  - Uchovávají veškerou informaci o minulosti, která je potřebná pro stanovení budoucího chování obvodu

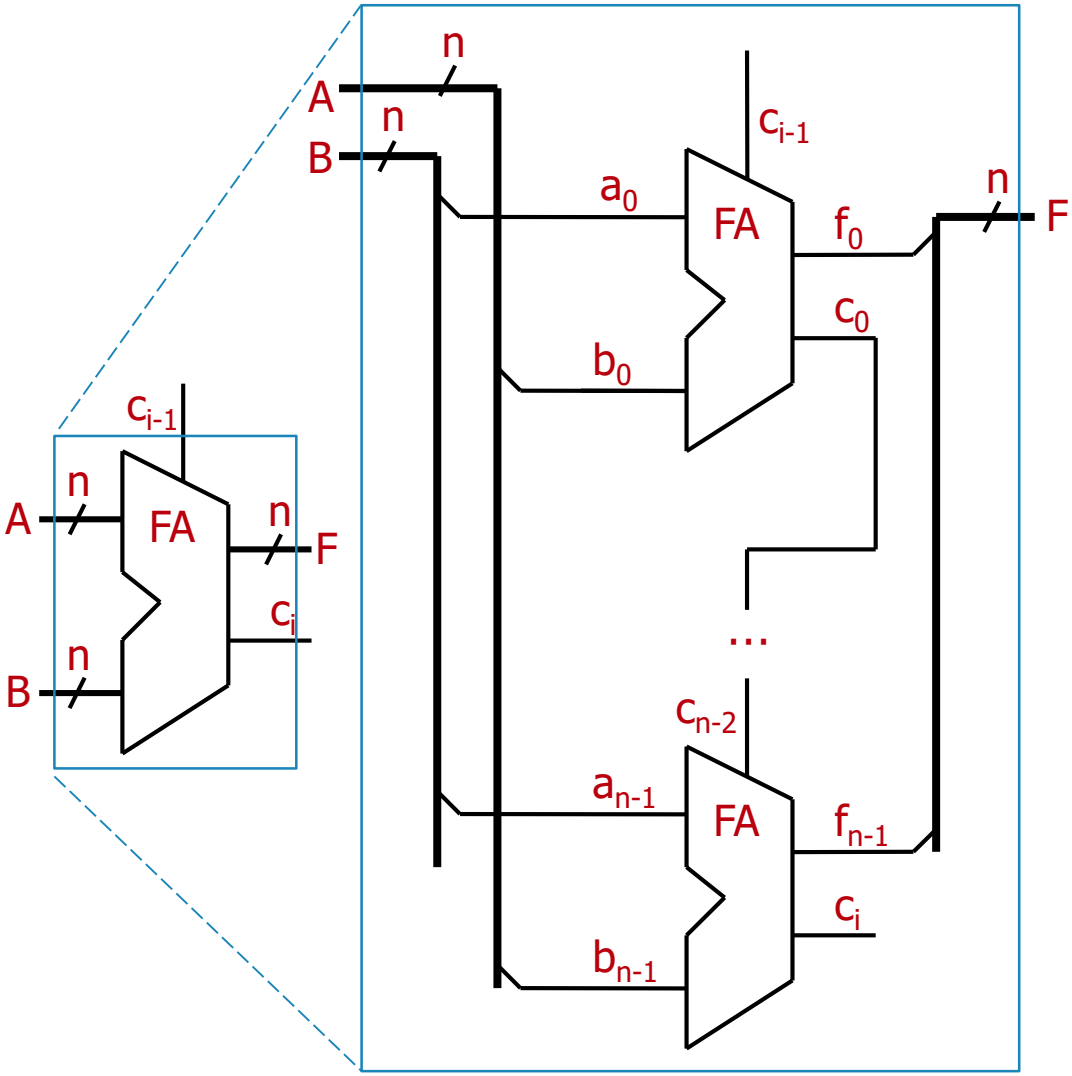


Dnešní systémy mají miliardy tranzistorů...

- Návrh složitých systémů
  - 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 je
  - 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)

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

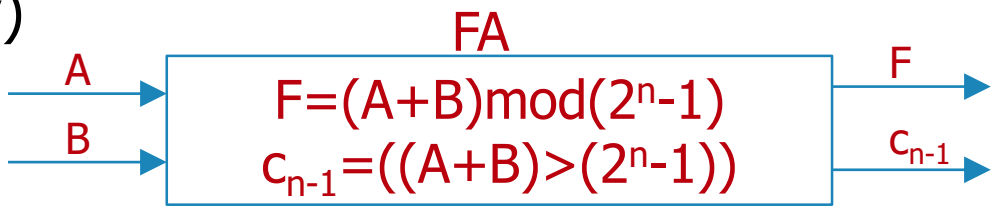


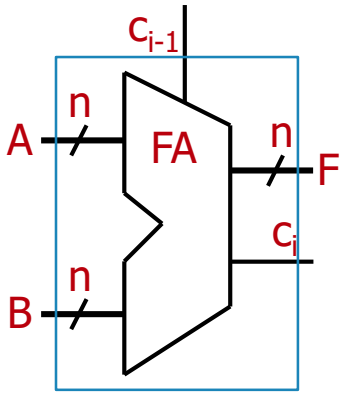
Popis systémů: chování - struktura

- 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

- 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 jednobitová sčítačka
  - Popsána algebraickými výrazy (popis chování) a jako logická síť (popis struktury)

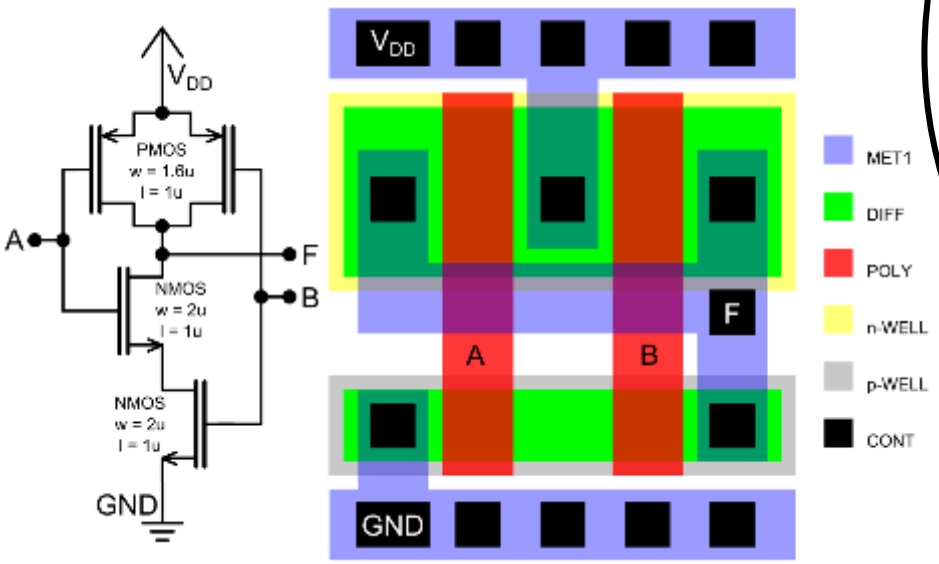


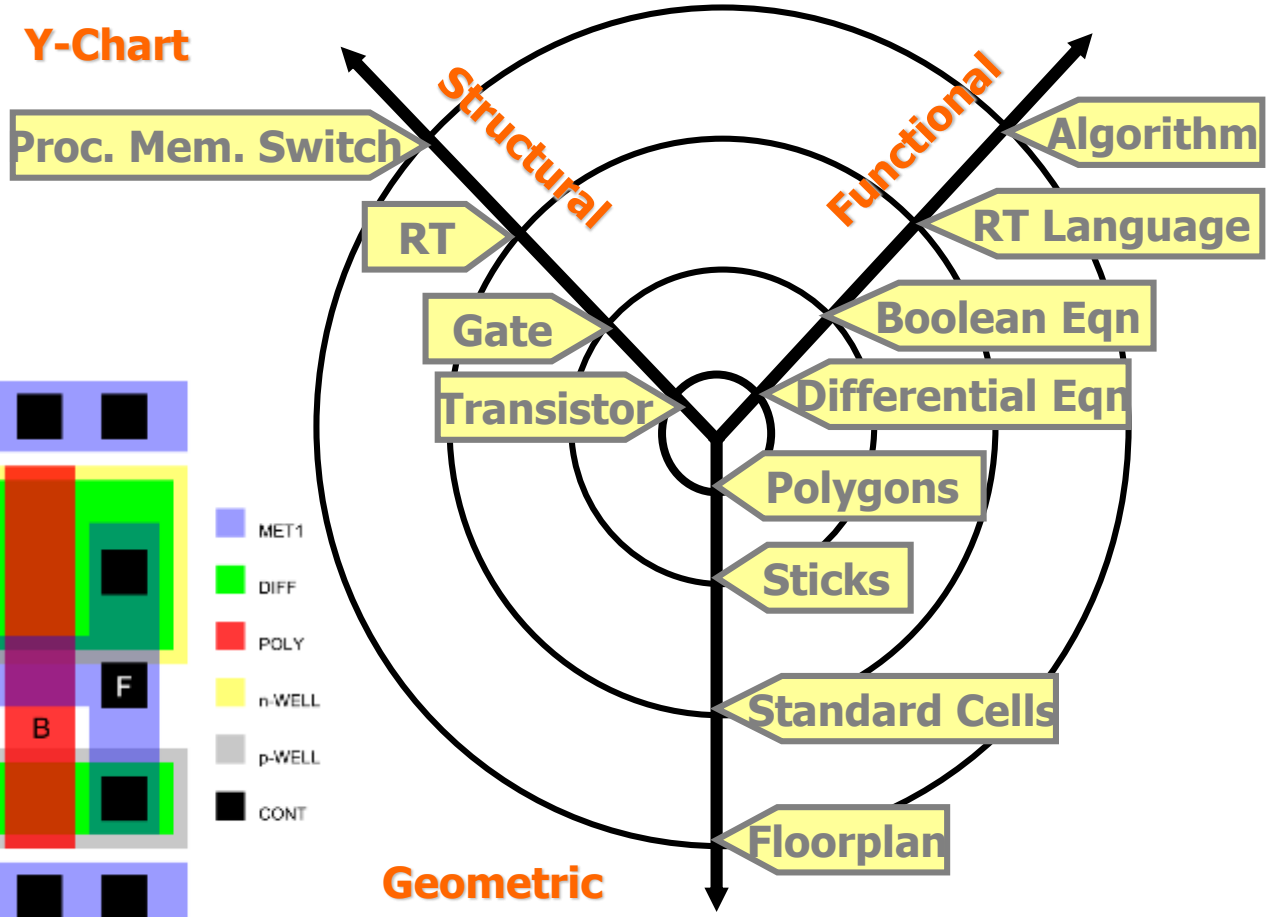


Úroveň popisu	Úroveň abstrakce	Množství detailů
Systém	Nejvyšší	Nejméně
Přenosy mezi registry	↑	↓
Log. obvody a členy	↑	↓
Tranzistory	↑	↓
Polovodičová podložka	Nejnižší	Nejvíce

Popis systémů: Úrovně abstrakce

- Analýza a návrh (syntéza) obvodů může být prováděna na více úrovních abstrakce
- Příklad: schéma – rozmístění a propojení na čipu

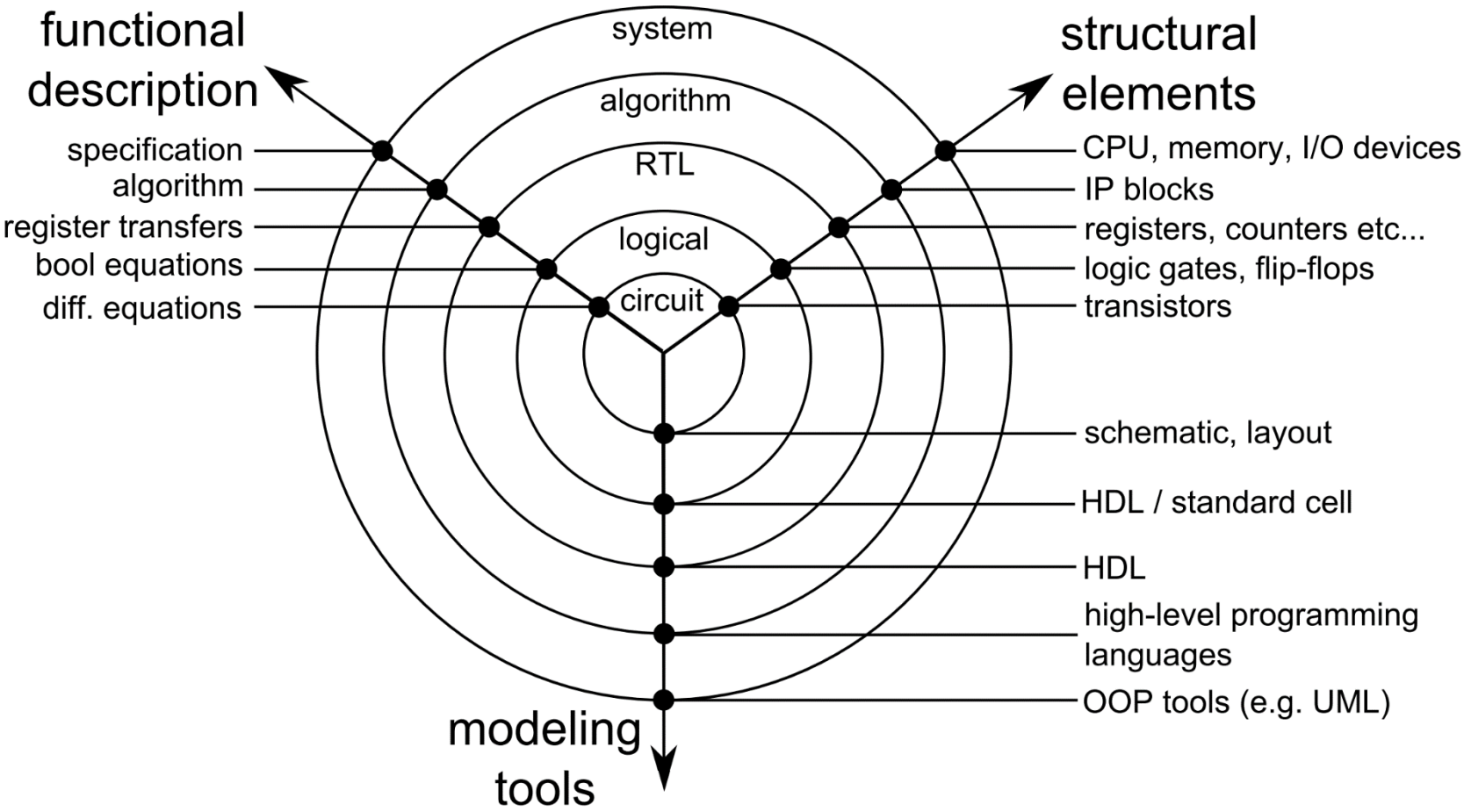




[Obrázek: K. S. Chatha: CSE 591 Hardware/Software Co-design, http://cse.asu.edu/7Ecse591b]

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: Slovní specifikace

- Pro člověka přirozený způsob specifikace funkce
  - Nejčastěji neformální slovní popis - např. zákazník požaduje: "navrhněte (číslicový) obvod, který spočte sumu všech členů dané posloupnosti"
- Daný slovní popis chování je nutno formálně zapsat (programovací jazyk, matematický výraz apod.)
  - Je třeba jednoznačně definovat vstupní a výstupní podmínky, určit obor hodnot, kterých mohou zpracovávané veličiny nabývat, definovat tok informace v systému a jednotlivé kroky výpočtu
- Prozatím neexistují nástroje pro automatizovaný návrh z neformálního slovního popisu

Popis systémů: Slovní specifikace - příklad

- AND
  - Např.: výstup má hodnotu "1", pokud mají všechny vstupy hodnotu "1"; jinak má hodnotu "0"
  - Nebo: Pokud je na kterémkoliv ze vstupů hodnota "0", tak výstup má hodnotu "0"; jinak má hodnotu "1"
- OR
  - Např.: Výstup nabývá hodnoty "0", pokud mají všechny vstupy hodnotu "0"; jinak má hodnotu "1"
  - Nebo: Pokud je na kterémkoliv ze vstupů hodnota "1", tak výstup má hodnotu "1"; jinak má hodnotu "0"
- NOT
  - Když vstup má hodnotu "0", tak výstup má hodnotu "1"; a naopak
- Lze slovně popsat počítač s miliardou komponent?

Přechod z popis chování na popis struktury

- Z formálního popisu chování je třeba navrhnout takovou výpočetní strukturu (logický obvod), která daný algoritmus implementuje
  - Návrh vhodné struktury (syntéza) se provádí ručně (náplň tohoto kurzu) či do značné míry automatizovaně (viz návazné kurzy, v praxi s ohledem na "time-to-market" nezbytnost)
- Existuje nekonečně mnoho výpočetních struktur, které implementují daný algoritmus (vícenásobná realizace)
  - Jejich vlastnosti určují rychlost výpočtu, příkon, rozměry, atd.
- Tvorba (výběr) určité struktury je centrálním problémem návrhu číslicových systémů

Popis chování: Matematicky

- Příklad:

$$S = \sum_{i=1}^{100} V_i;\ \{S \in \mathrm{N}^+ \mid 0 \le S \le 900\}, \{V_i \in \mathrm{N}^+ \mid 0 \le V_i \le 9\}$$

  - Uvedený výraz představuje popis chování obvodu
- Použití matematického popisu jako podkladu pro fyzickou implementaci příslušného obvodu (výrobu)
  - V současné době nejsou běžně dostupné nástroje, které by provedly efektivní (rychlost vs. příkon vs. cena) implementaci obvodu z čistě matematického popisu chování plně automatizovaně (bez úzké asistence člověka – návrháře), nebo jen v omezené formě (např. syntéza z jazyka Matlab)
  - Člověk musí, na základě např. slovního či matematického popisu, vytvořit popis na nižší úrovni abstrakce (nejčastěji v programovacím jazyku), který pak bude sloužit jako podklad pro výrobu příslušného obvodu

Popis chování a struktury: Programovací jazyky

- Dnes nejrozšířenější způsob popisu systémů
  - Vhodné jak pro popis chování, tak struktury
- Vycházíme nejčastěji ze slovního či matematického popisu
  - Popis chování obvodu v programovacím jazyku lze vytvořit pomocí vhodných jazykových konstrukcí, jako jsou např. přiřazení, porovnání, smyčky, ap.
  - Postup je dnes společný návrhářům software i hardware
- Příklad
  - Pozn.: Je třeba též definovat obor hodnot jednotlivých proměnných a vstupní podmínky

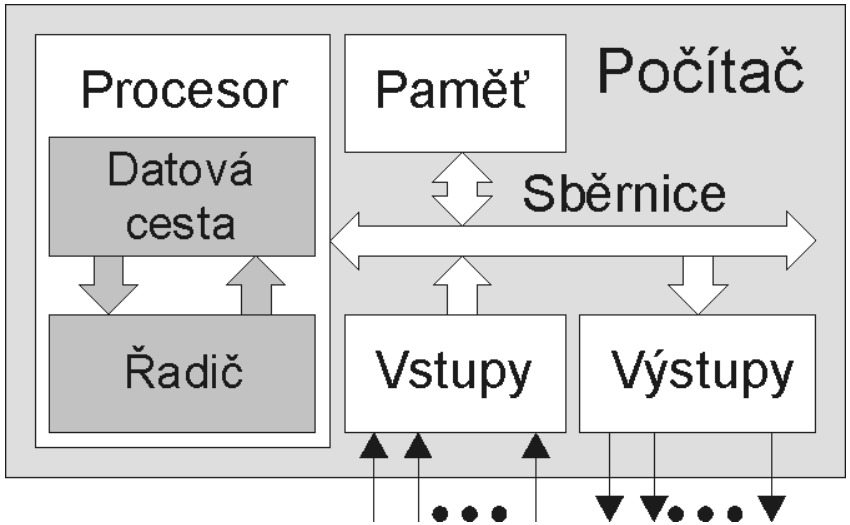
$$S_0 \leftarrow 0; \qquad vynulujregistr$$

$$\underline{for}\ \ i = 1\ \ \underline{to}\ \ N\ \ \underline{do}; \qquad provšechna\, V_i$$

$$(S_{i+1} \leftarrow S_i + V_i); \qquad akumuluj$$

Úroveň popisu: Systém

- Nejvyšší úroveň popisu
  - Systém je definován jako jedna či více vzájemně spolupracujících komponent (funkčních jednotek)
  - Chování každé komponenty je popsáno bez specifických implementačních detailů
- Např. stolní počítač
  - Systém skládající se z procesoru, paměti, sběrnice a periferních jednotek jako jsou klávesnice, tiskárna, displej ap.

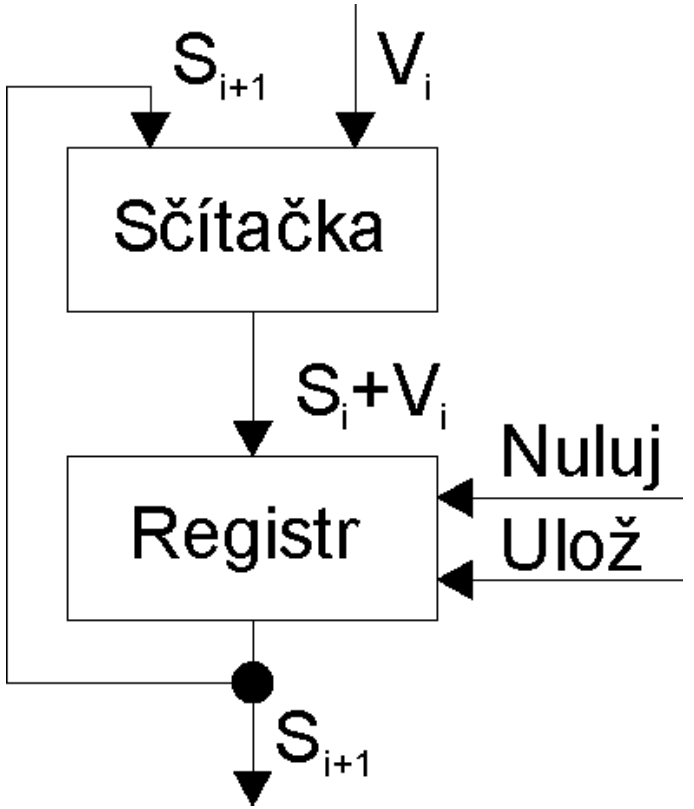


Úroveň popisu: Přenosy dat mezi registry (RTL)

- Register Transfer Level (RTL)
  - Úroveň přenosu mezi registry
- Jednotlivé funkční jednotky (např. sčítačky, násobičky…) a úložiště dat (tzv. registry) mezi sebou propojeny pomocí signálů (vodičů)
  - Informace je v obvodu vhodným způsobem zpracovávána příslušnými funkčními jednotkami a přenášena mezi jednotlivými registry
- Příklad: Slovní popis chování obvodu na RTL přenosů mezi
  - "*Navrhněte obvod, který nejprve signálem Nuluj vynuluje pracovní registr S. Následovně se pomocí signálu Ulož, aktivovaného celkem N×, ukládá výsledek součtu hodnot jednotlivých členů posloupnosti s předchozí (akumulovanou) hodnotou uloženou v registru S. Výpočet končí po N krocích, kdy je v registru S (nazýván též akumulátor) uložen výsledek.*"

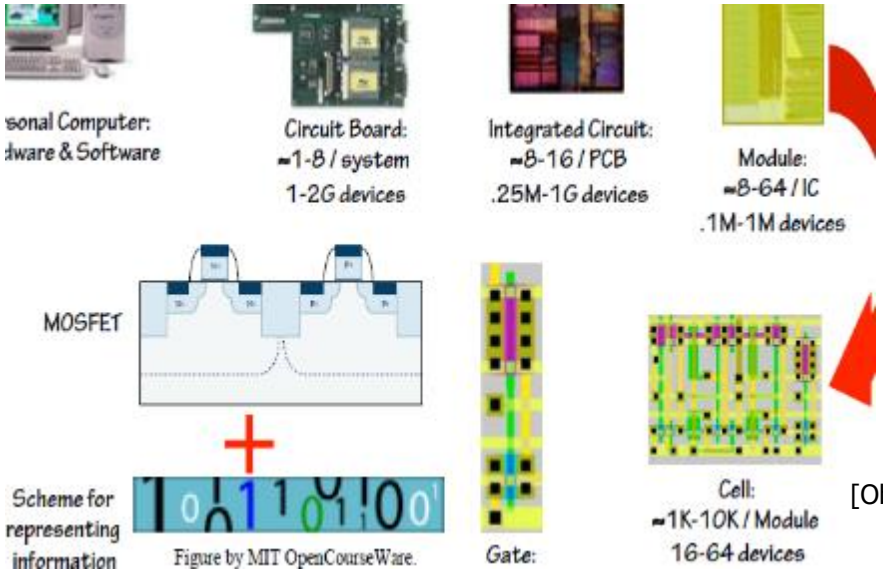
Úroveň popisu: Přenosy dat mezi registry (RTL)

- Stejný obvod lze též popsat strukturou – např. 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 sčítání

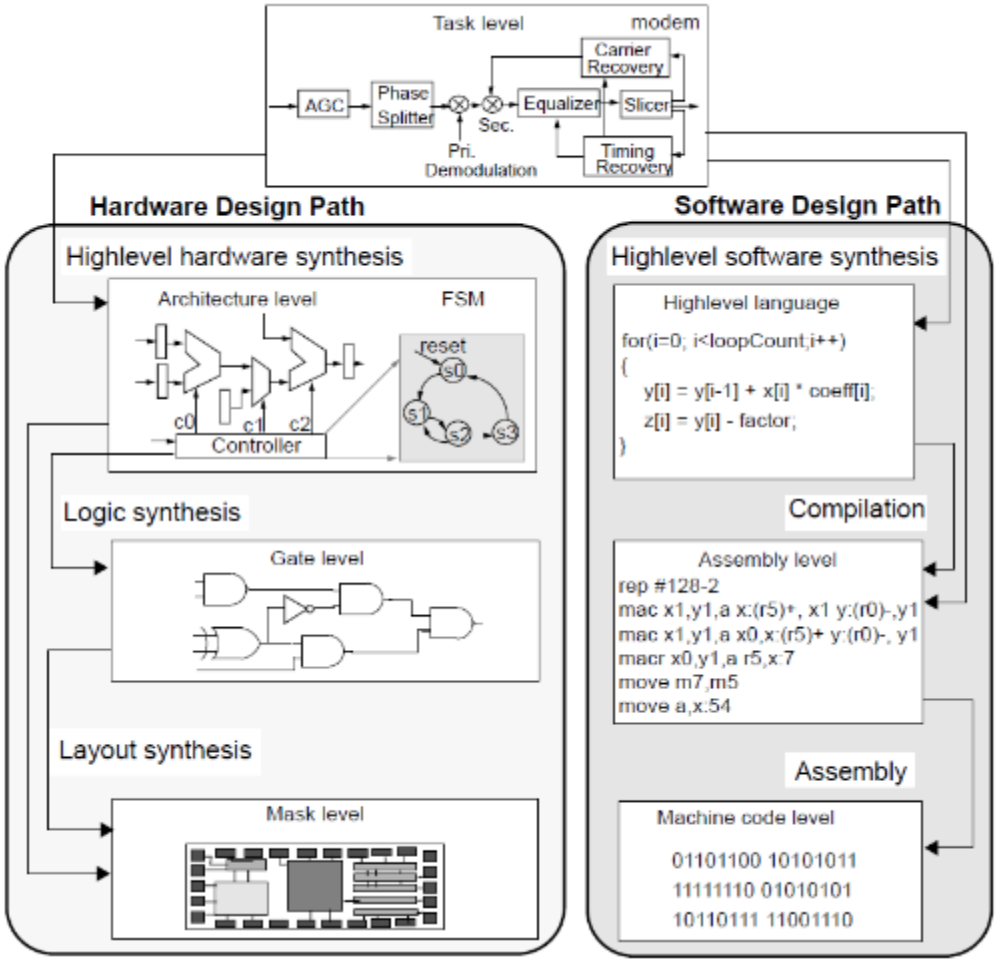


Různé úrovně a způsoby popisu systému: Příklady

- 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

Co je to algoritmus?

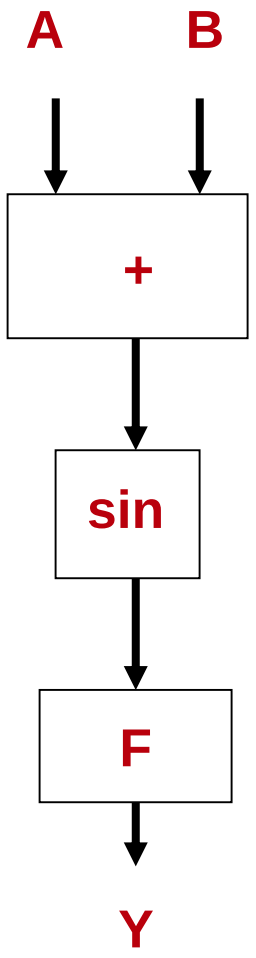
- Při řešení úloh jde tedy o návrh algoritmu a jeho následnou fyzickou implementaci
- Algoritmus
  - Konečná uspořádaná množina úplně definovaných pravidel pro vyřešení nějakého problému
- Intuitivně
  - Postup, který nás dovede k řešení úlohy
- Formálně
  - Přesně definovaná konečná posloupnost příkazů (kroků), jejichž prováděním pro každé přípustné vstupní hodnoty získáme po konečném počtu kroků odpovídající výstupní hodnoty [z kurzu Základy programování]

Implementace algoritmu

- V software (SW)
  - Výpočet běží na univerzálním procesoru
    - Datové struktury
    - Řídicí struktury – sekvence, rozhodování, iterace
- V hardware (HW)
  - Specializované obvody pro konkrétní úlohu
    - Datové struktury – registry, paměti apod.
    - Řídicí struktury – řadič řídí funkční jednotky a datovou cestu
  - Urychlení, nižší cena a spotřeba pro vhodnou třídu aplikací
    - Ne univerzální stroj, ale aplikačně specifický HW – implementuje se jen to, co je nezbytně třeba pro danou funkci
- Limitující faktory
  - Výkonnost, latence, příkon, kapacita paměti, cena a rychlost návrhu atd.

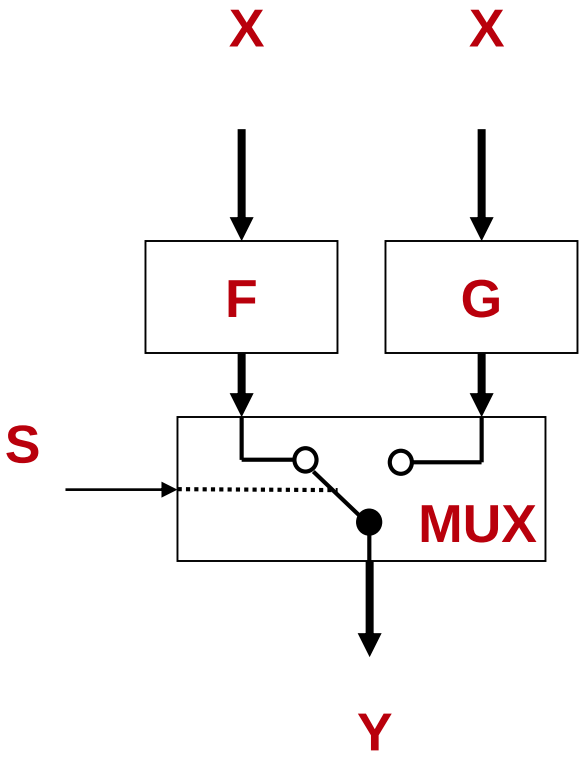
Implementace algoritmu: SW vs. HW - Sekvence

```
A = B + C;
D = sin(A);
Y = F(D);
```



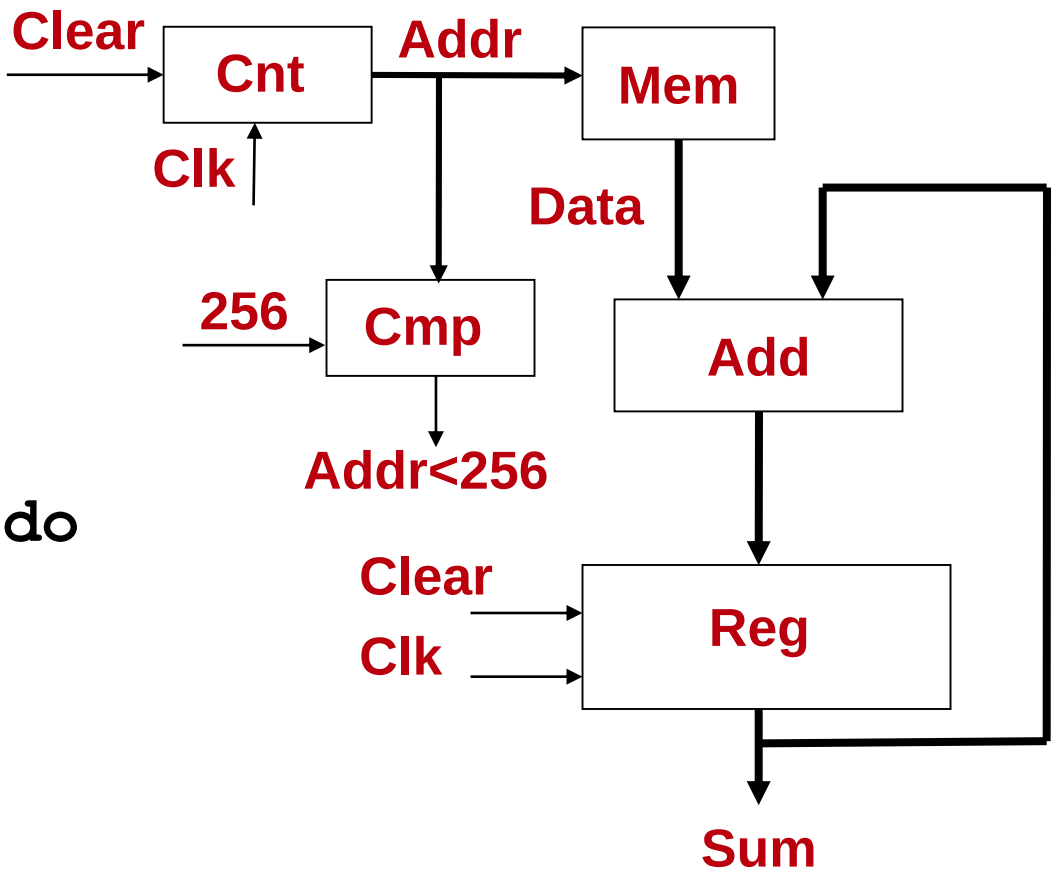
```
if (S == 0) then Y = F(X)
else Y = G(X);
```

- Poznámka:
  - V SW implementaci se spočte buď funkce F, nebo G. Zpracování informace v SW je přirozeně sekvenční.
  - V HW implementaci se spočtou obě funkce F a G. Přepínačem (multiplexor – MUX) se pak podle hodnoty na vstupu S (0, 1) vybere výsledek funkce F, nebo G. Zpracování informace v HW je přirozeně paralelní.



- Součet členů posloupnosti

```
if Clear than
{
  Addr = 0;
  Sum = 0;
};
while (Addr < 256) do
{
  Data = Mem[Addr];
  Sum = Sum + Data;
  Addr++;
};
```



Implementace algoritmu: HW vs. SW

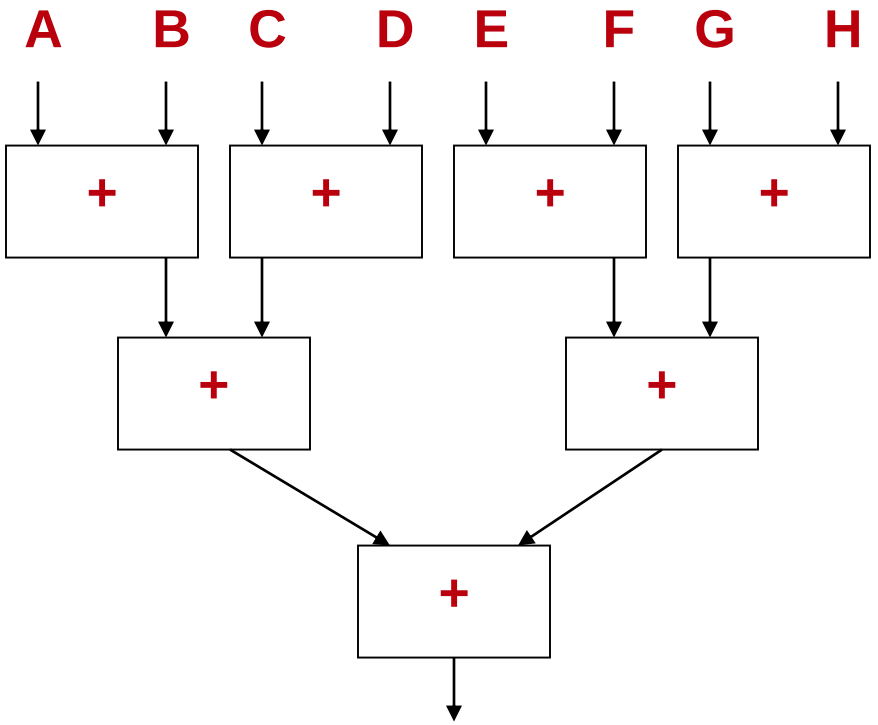
- Výpočet je v HW efektivnější (rychlost, příkon) než v SW
  - Lze realizovat speciální kódování dle potřeby dané úlohy
    - Příklad: aritmetické operace v kódu zbytkových tříd jsou extrémně rychlé
  - Lze realizovat speciální výpočetní jednotky dle potřeby dané úlohy – např. rychlá Fourierova transformace (FFT)
  - Paralelní zpracování (násobné výpočetní jednotky)
  - Zřetězené (pipeline) zpracování, atd.
- Návrh SW je efektivnější (cena, doba návrhu) než HW
  - Návrh a výroba HW jsou "drahé"
    - Složité - časově a finančně náročný návrh, nutno amortizovat výrobu atd.
  - Programování SW je "levné"
    - Uživatel pouze formuluje algoritmus, nezabývá se návrhem a výrobou HW systému

Výpočet v SW a HW (paralelní zpracování)

- Výpočet: S = A + B + C + D + E + F + G + H

```
S = A + B;
S = S + C;
S = S + D;
S = S + E;
S = S + F;
S = S + G;
S = S + H;
```

7 kroků



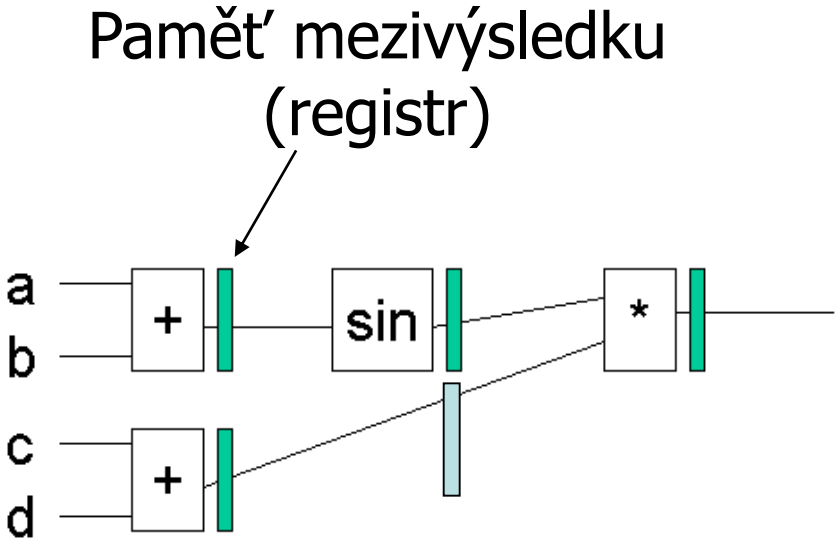
3 kroky

Výpočet v SW a HW (zřetězené zpracování)

- Výpočet: F[i]=(c[i]+d[i])*sin(a[i]+b[i]), pro i=1..100

```
for (i=1; i<=100; i++)
{
  pom = a[i]+b[i];
  pom1 = sin(pom);
  pom2 = c[i]+d[i];
  F[i] = pom1 * pom2;
}
```

Celkem: 100 x 4 = 400 kroků



První výsledek za 3 kroky
2. až 99. výsledek za 1 krok
Celkem 102 kroků
Zrychlení 4×