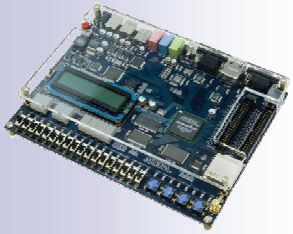
Struktury počítačových systémů



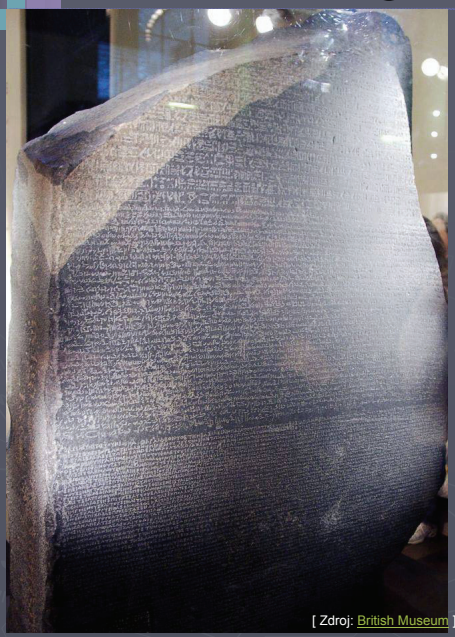
Přednáší: Richard Šusta

Verze 1.2 - opravené prohozené výsledky
ErrorOdd9a *na snímku 38 a 39*

1.1 - opravena chyba na snímku 48
+ vynechány neodpřednášené části

ČVUT-FEL v Praze – kód předmětu A0B35SPS

Omluva



[Zdroj: British Museum]

Většina snímků nebude v češtině

** neexistují vždy **jednotné** české ekvivalenty - známé budu však uvádět v závorkách (cz:...)*

** předmět se prvně přednášel v roce 2009 pro anglické studenty a znovu je v programu pro zahraniční studenty,...*

Přednášející

Richard Šusta
richard@susta.cz
+420 2 2435 **7359**

Stránky předmětu

https://moodle.dce.fel.cvut.cz/course/view.php?id=5

https://moodle.dce.fel.cvut.cz/ -> Kybernetika a robotika / Bakalářské předměty

https://dce.fel.cvut.cz/ -> E-kurzy systému Moodle
-> Kybernetika a robotika / Bakalářské předměty

http://susta.cz/fel/ nebo www.susta.cz/fel
-> Struktury počítačových systémů
-> Altera FPGA
-> Kurz FPGA

Obsah přednášky

1. *O struktuře počítačových systémů sekvenční versus paralelní*
2. *Geneze hradla na příkladu ...a proč se to máte učit*
3. *Logická funkce - matematický zápis versus realizace*
4. *Booleovská algebra*

Program SPS

O struktuře

Jednoduché číslicové obvody

- Kombinační
- Sekvenční obvody
- Realizace automatů

CPU

- Řadiče
- ALU, Instrukce
- Paměť

Počítačový systém

- Procesor NIOS
- Sběrnice
- Vstupy a výstupy
- Přerušení

O realizaci

Programování

Schéma
+později VHDL

Vývojové prostředí

Quartus II, NIOS

Hardware

Altera Cyclone II
DE2 Board

Umístění předmětu

1. ročník

- LGR: Logika a grafy

OI+ DMA: Diskrétní matematika

2. ročník

zimní

SPS: Struktury počítačových systémů

OI+ JAG: Jazyky, automaty a gramatiky

letní

- **APO - Architektura počítačů**

Volitelné

- APH Aplikace programovatelných hradlových polí
- DIT Digitální technika
- MIK Mikrokontroléry
- PPS Průmyslové počítačové systémy

Předběžný přehled předmětu

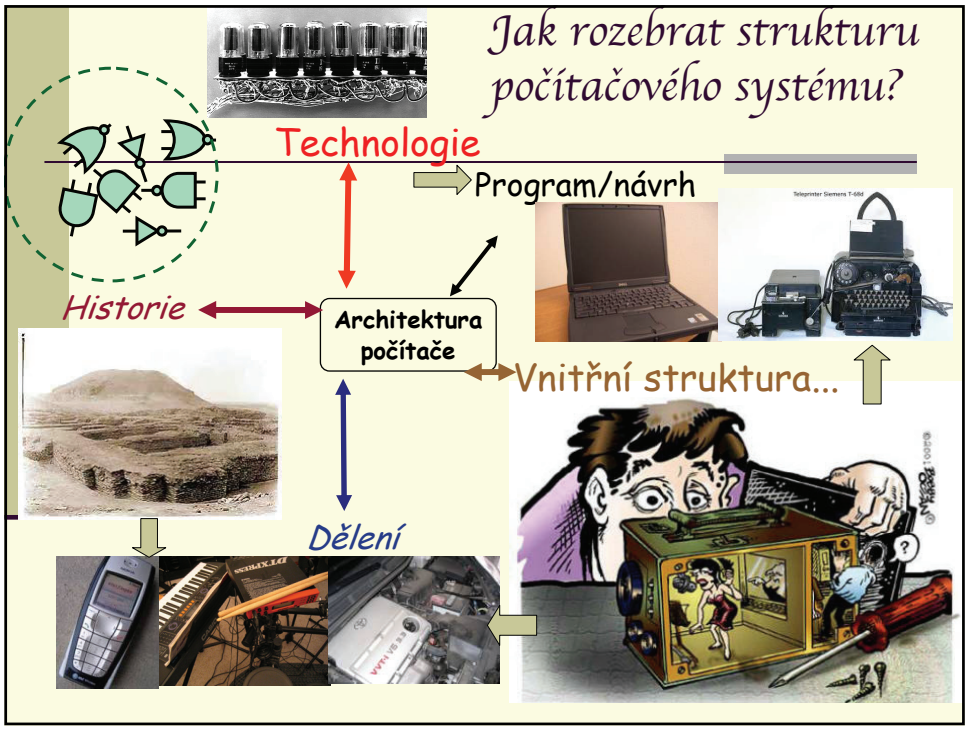
	Hlavní téma přednášky	Cvičení
1	Logická funkce;	Organizace
2	Sekvenční obvody, čítače	Vývojové prostředí Quartus II
3	Minimalizace, hazardy.	Projektu 1 (čítače).
4	RTL syntéza a popisy v jazyce VHDL.	Projektu 2 (světelný had).
5	Serializace (RS232, PS-2, VGA).	Samostatná práce
6	Řadiče, příklad řadič LCD a klávesnice.	Projekt 3 (digitální hodiny).
7	Operace procesoru.	Samostatná práce.
8	Struktura procesoru. Cykly procesoru.	Projekt 4 (VGA).
9	FPGA procesorový systém NIOS.	Samostatná práce
10	Sběrnice, tvorba systému.	Samostatná práce.
11.	Paměti. Přerušení.	Projekt 5 (procesorový systém NIOS)
12.	Interfacing.	Samostatná práce.
13.	Obvody FPGA a ASIC.	Samostatná práce.

14. týden odpadne na svátcích

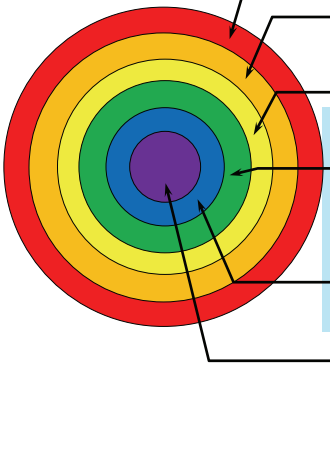
Literatura

Přednášky + cvičení

1. Enoch O. Hwang: Digital Logic and Microprocesor Design with VHDL, Electronix 2004, 513 s.
2. Bayer, J. - Hanzálek, Z. - Šusta, R.: *Logické řízení*. 2. vyd. Praha: ČVUT 2008. 270 s. ISBN 978-80-01-04106-2.
3. Bayer, J. - Šusta, R.: *Logické řízení - cvičení*. 1. vyd. Praha: ČVUT 2008. 208 s. ISBN 978-80-01-04105-5.



Vnitřní struktura



- Aplikační program
- Operační systém
- Ovladače zařízení (driver)
- **SPS**
  - Komponenty počítače: I/O zařízení, procesor(y), paměti, sběrnice
  - Obvody logické a analogové
- Prvky uvnitř IO

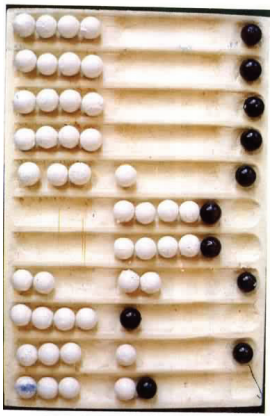
Computer

A long history of TECHNOLOGY

Digital computers -> Simple +/- tasks

- **Calculus** (cz:kalkul) from Greek name for **pebbles** (*cz: oblázek, valounek*)
- **Abacus**, pl. **abaci** or abacuses (*cz:abakus, abak, počítadlo*)

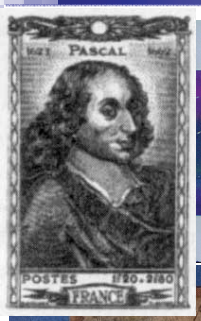
the first appearance in Sumer, 2700-2300 BC

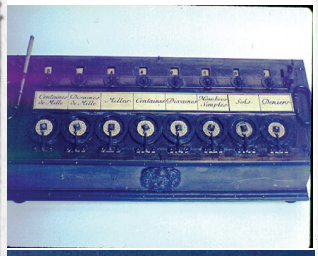


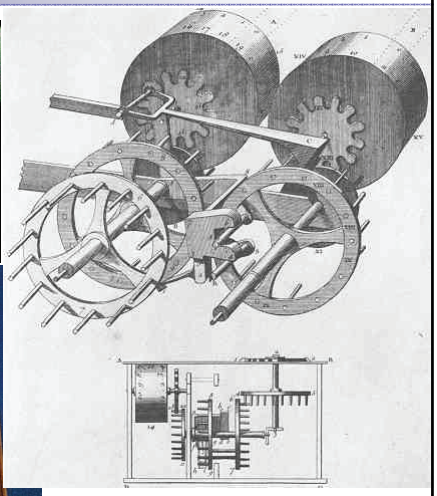
an adapted abacus is used until today

[Photo: Martina Nicolls in Georgia (*cz:Gruzie*)]

Blaise Pascal's Pascaline (1645) - Simple +/- operationss



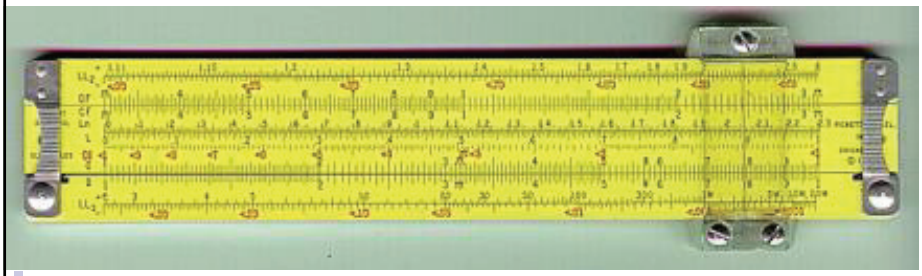




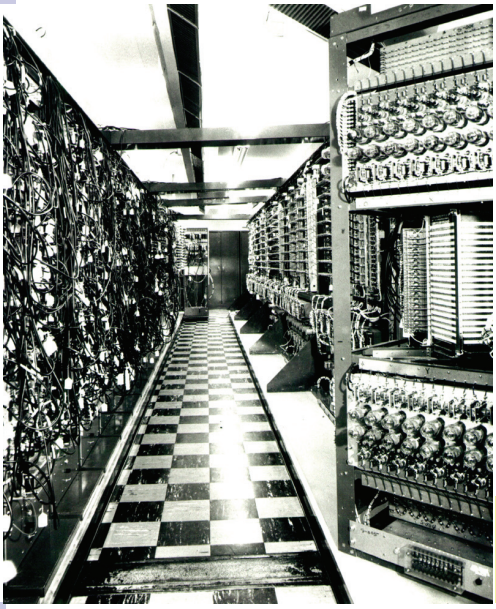
Ancient analog computers -> Complex tasks

Astrolabe
(cz:astroláb, -u, m.; astrolabium, -ia, s.)
150 BC Greek astronomer and geographer Hipparchos z Bithynie

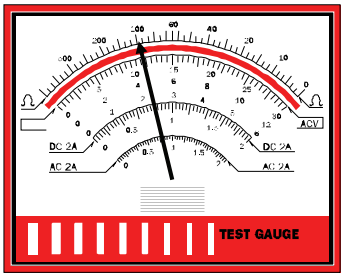
Slide rule *(cz:logaritmické pravítko)*
Oughtred (1621) and Schickard (1623)



1943 Analog computer Whirlwind from MIT



Fast, but inexact



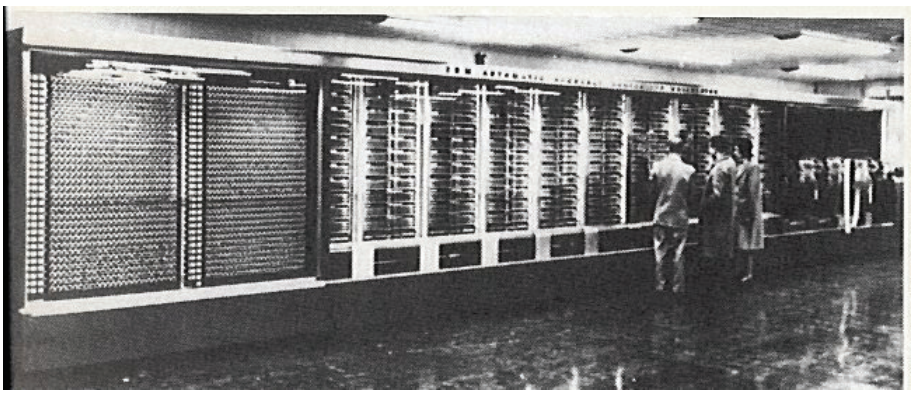
About 100

103.5

103.5

Digital has only one interpretation

1944 Mark I Calculator



Weight 4500 kg, built from 800 km wires and 3500 relays,
Memory: 72 words with 23 digits

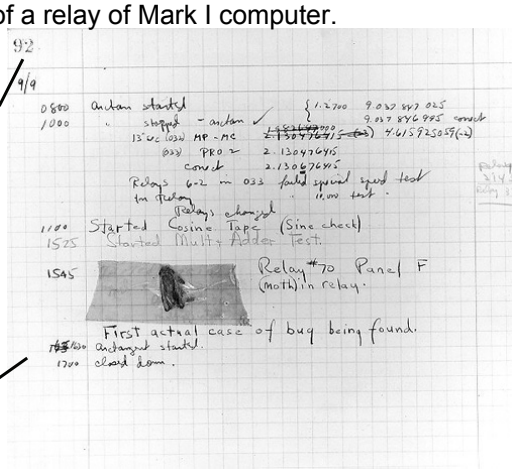
The First Computer Bug

- Grace Murray Hopper found the first computer bug

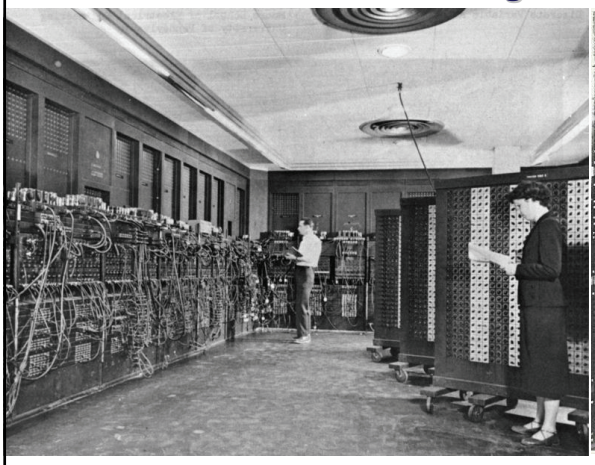
  beaten to death in the jaws of a relay of Mark I computer.

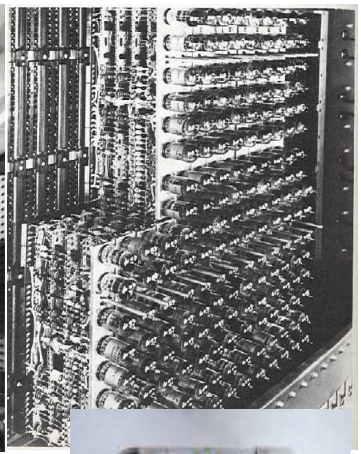
Numbered pages for USA patents

Lab book!!



1946 The ENIAC:
Electronic Numerical Integrator and Computer

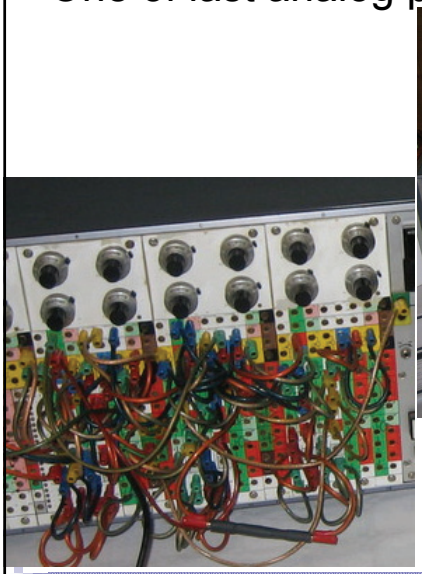


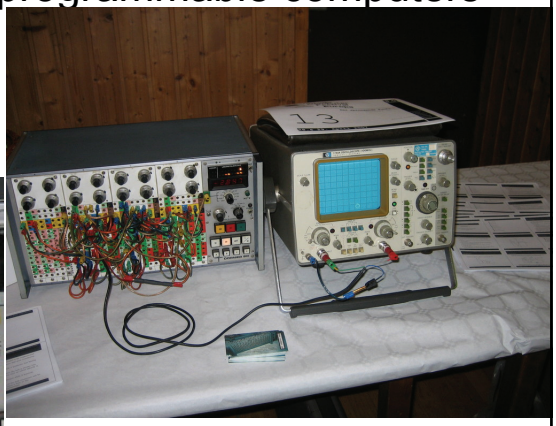


Weighed over 30 tons, and stored a maximum of twenty 10-digit decimal numbers

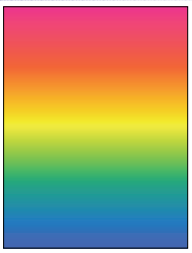
Analog Computers ???

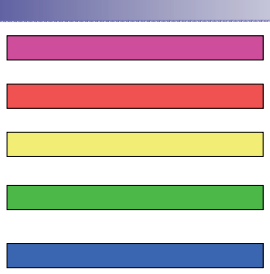
- One of last analog programmable computers

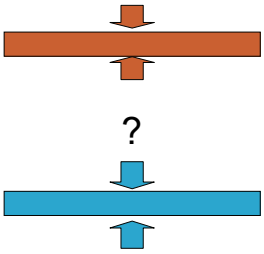


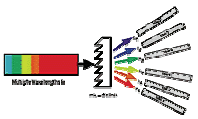


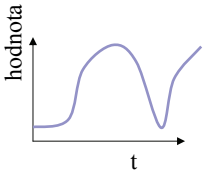
Analog offers Continuous Spectrum
Digital offer distinct Steps

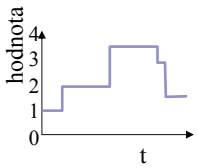


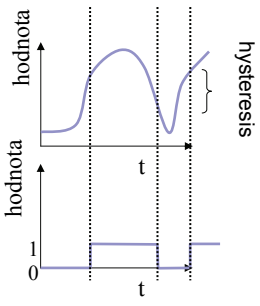






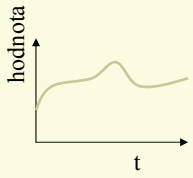




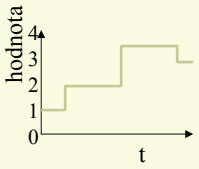


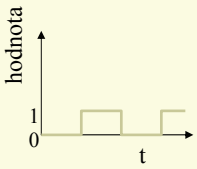
Pojmem signál v SPS…

- Analogový signál
  teoreticky nekonečně mnoho hodnot



- Digitalní (číslicový) signál
  konečný počet hodnot



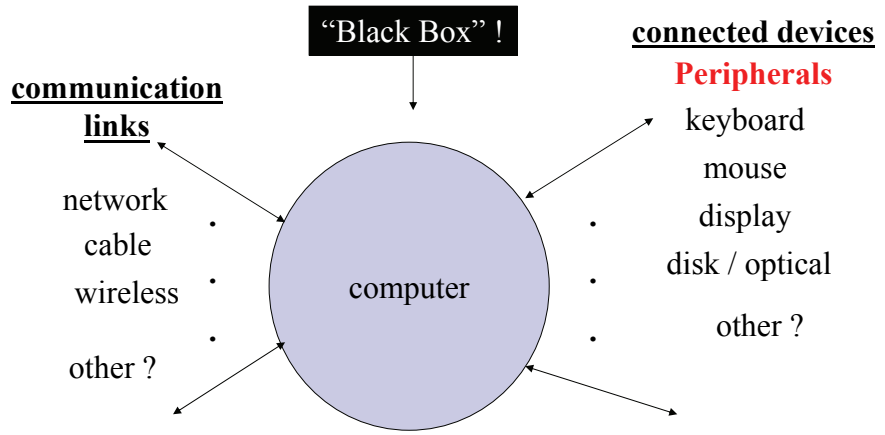


- Binární signál
  – digitální signál o 2 hodnotách značených
  '0' a '1' nebo 'low' a 'high' či 'true' a 'false' ...
  -> Java typ boolean
  = one *b*inary dig*it* = *bit*

Sequential Computer System

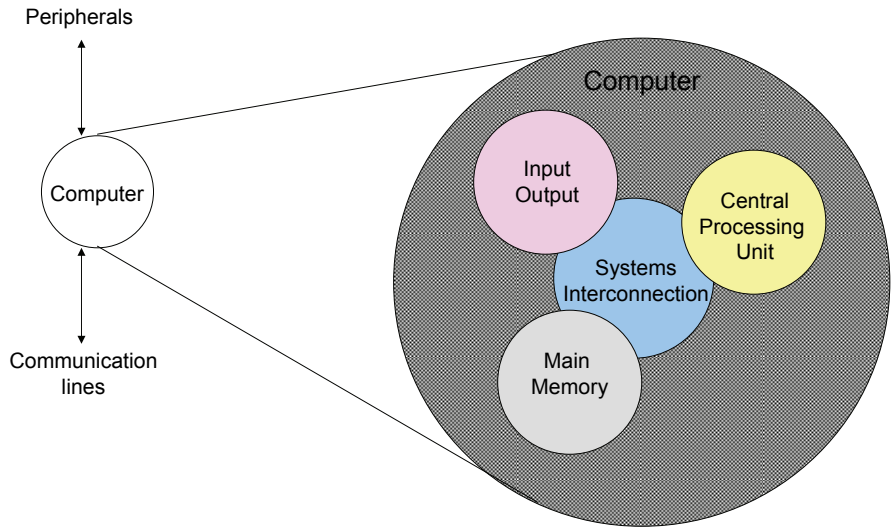
and its interior

What should I know already about Digital Computer ?



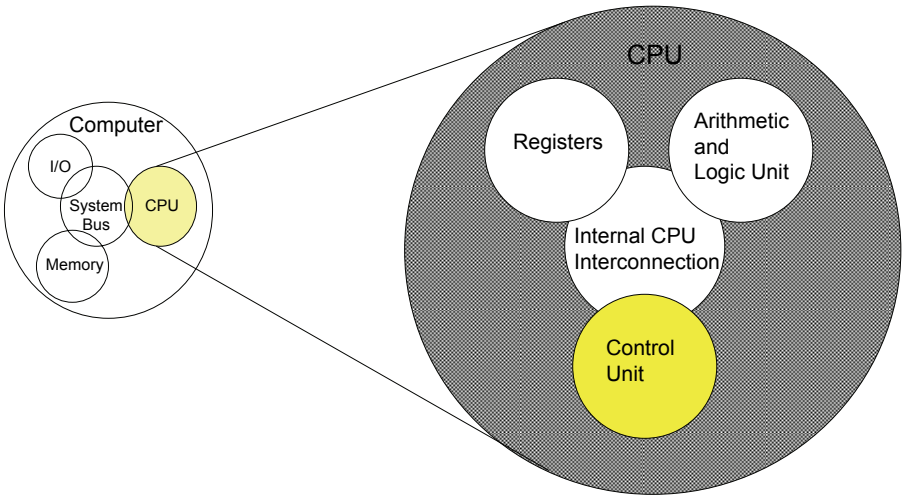
Structure - Top Level

More Black Boxes!

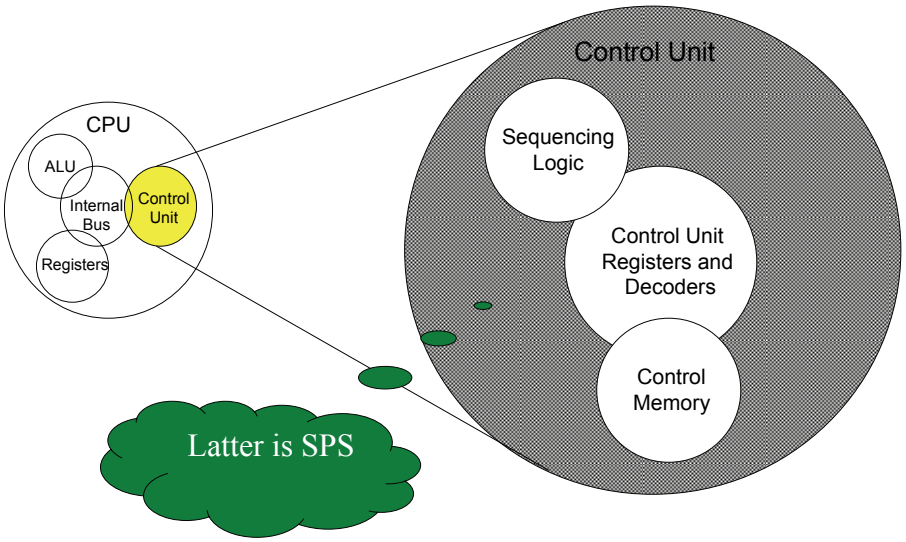


Structure - The CPU

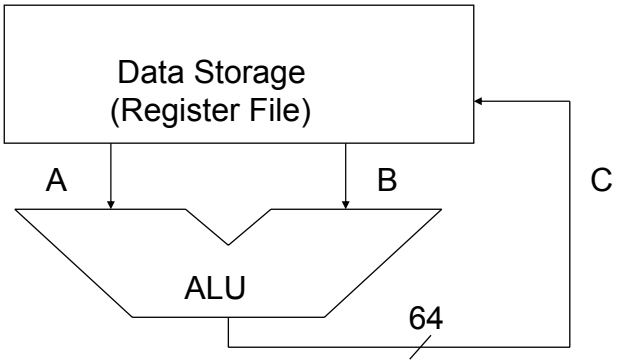
More More Black Boxes!



Structure - The Control Unit



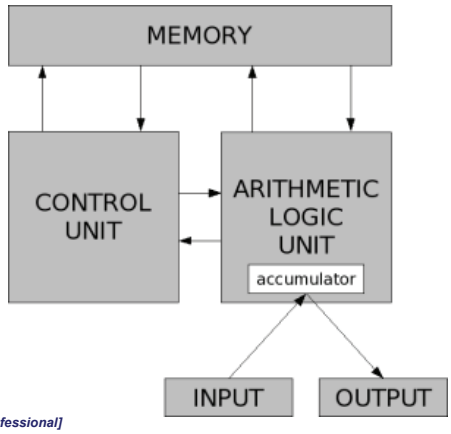
Microprocessor-based Systems



- Generalized to perform many functions well.
- Operates on fixed data sizes.
- Inherently sequential.

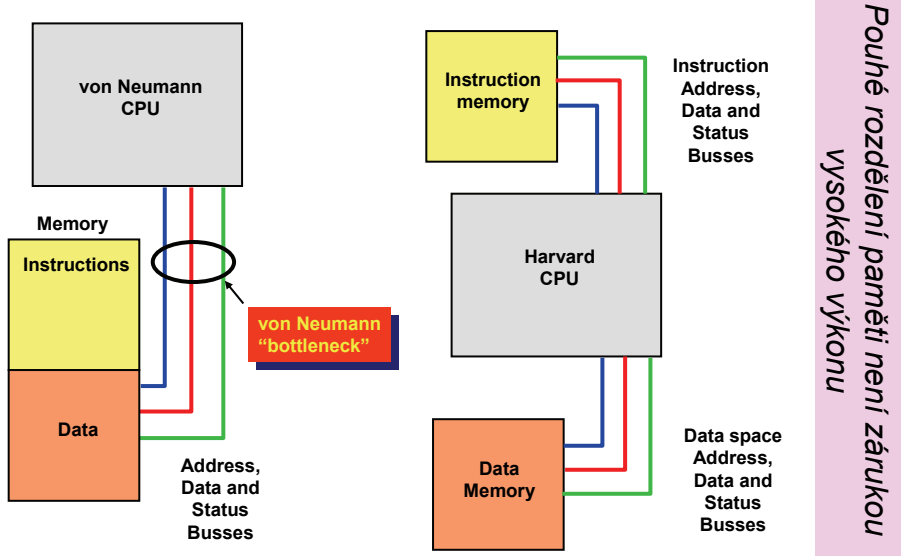
Von Neumann Machine, 1945

- Stored-program Model
  - Both data and programs are stored in the *same* main memory
- Sequential Execution
- Memory, Input/Output, Arithmetic/Logic Unit, Control Unit



[Arnold S. Berger: Hardware Computer Organization for the Software Professional]

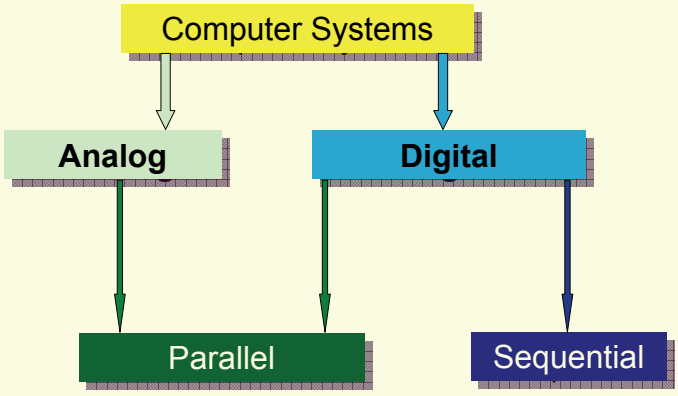
von Neumann and Harvard Architectures



Pouhé rozdělení paměti není zárukou vysokého výkonu

[Arnold S. Berger: Hardware Computer Organization for the Software Professional]

A Classification of Computer Systems

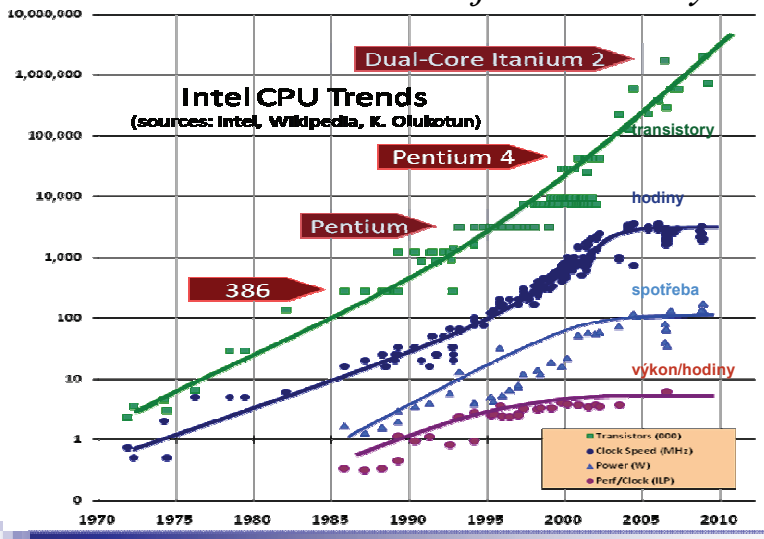


faster, bigger *slower, smaller*

... the comeback to available technology

Moore Law

- **Gordon Moore**, spoluzakladatel Intelu, v roce 1965 prohlásil :
  "*Počet transistorů se zdvojnásobí každým rokem*"

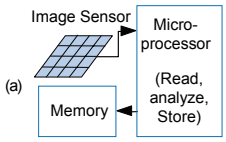


Logická funkce

realizace

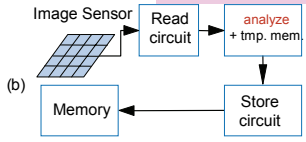
Příklad vlastního systému…

1. Procesor provádí operaci příliš pomalu
2. Realizujeme výpočetní operace v hardwaru
3. Urychlíme ho realizací **jen části** výpočetních operací v hardwaru, který pracuje paralelně



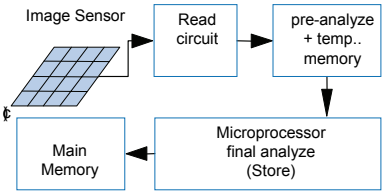
Sice rychle naprogramované, ale pomalé - 2 s na snímek

Rychlé, ale složité



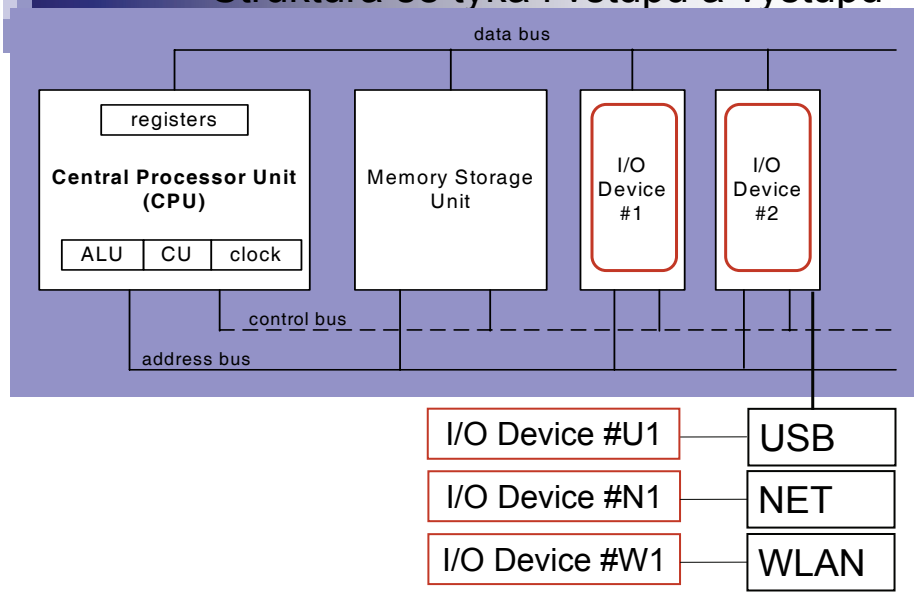
Zpracuje obraz za 10 ms, ale obtížný návrh

Kompromis



Zpracuje obraz za 50 ms a má snadný návrh z hotových bloků

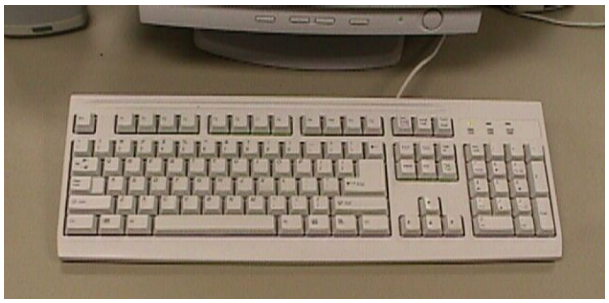
Struktura se týká i vstupů a výstupů

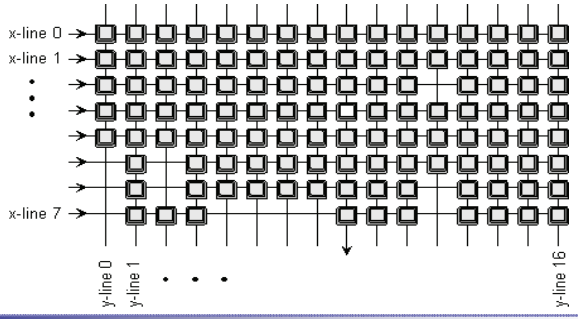


Evample

Design parity checker for keyboard

Keyboard scan matrix



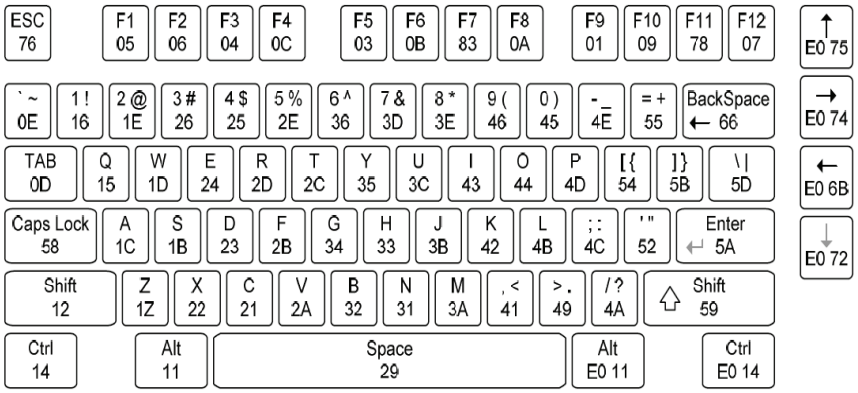


Hexadecimální scan kód = x-y souřadnice

8 datových bitů

lichá parita	P	7	6	5	4	3	2	1	0

***Lichá parita** (eng:odd parity) = zapíše se 1 nebo 0 do nejvyššího bitu, aby přenášeném 9-bitové slovo mělo lichý počet jedniček*



Parita in C++

```
// Is an error in odd parity of 9 lower bits
bool ErrorOdd9a(int data)
{   bool odd = true;
    for(int mask=1; mask < 0x100; mask=mask*2)
          {  if((data & mask) !=0 ) odd = !odd;  }
    bool bit8 = (data & 0x100) != 0;
    return bit8 != odd;
}
int _tmain(int argc, _TCHAR* argv[ ])
{   bool error1=ErrorOdd9a(0);          // true
    bool error2=ErrorOdd9a(1);          // false
    bool error3=ErrorOdd9a(0x101);      // true
    bool error4=ErrorOdd9a(0x1FF);      // false
    return 0;
}
```

Parita in C++

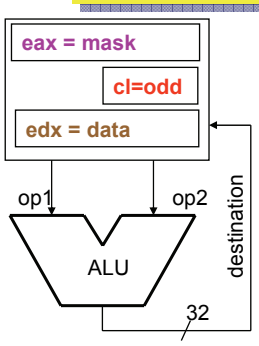
```
// Is an error in odd parity of 9 lower bits
bool ErrorOdd9(int data)
{   bool odd = true;
    for(int mask=1; mask <= 0x100; mask *=2)
          {  if( (data & mask) !=0 ) odd = !odd;  }
    return odd;
}
int _tmain(int argc, _TCHAR* argv[ ])
{   bool error1=ErrorOdd9a(0);              // true
    bool error2=ErrorOdd9a(1);              // false
    bool error3=ErrorOdd9a(0x101);          // true
    bool error4=ErrorOdd9a(0x1FF);          // false
    return 0;
}
```


Parita in C language -> sequential operation

C program

```
bool ErrorOdd9(int data)
{   bool odd = true;
    for(int mask=1; mask <= 0x100; mask *=2)
          {  if( (data & mask) !=0 ) odd = !odd;  }
    return odd;
}
```

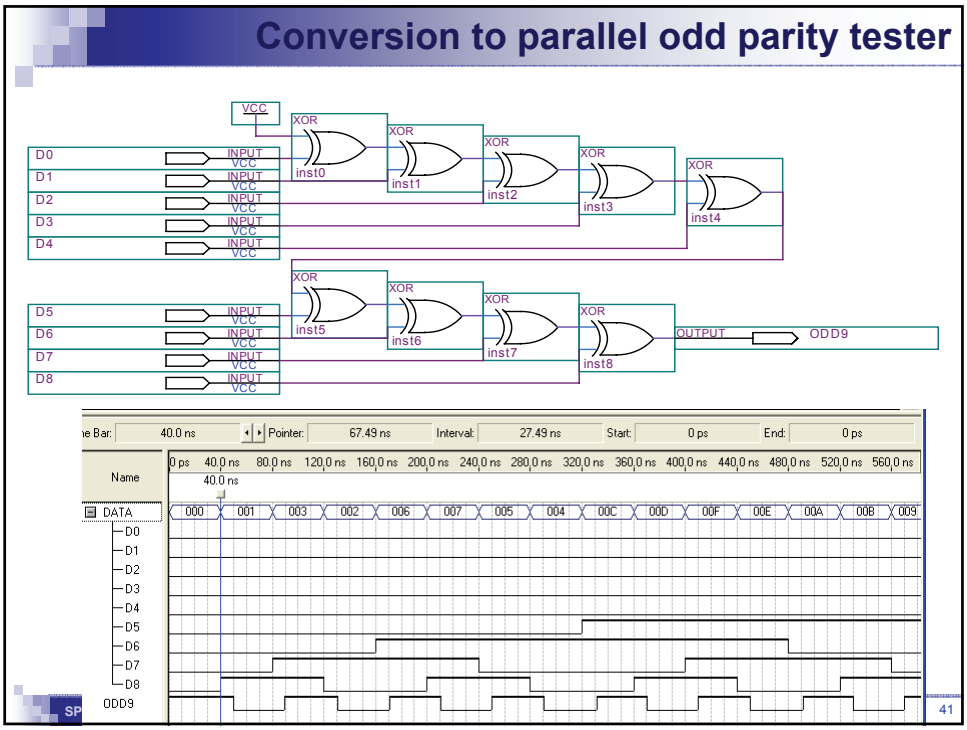
Assembler x86




```
/*....*/
0401008   mov    cl, 1;          // bool odd = true;
040100D   mov    eax, 1;         // int mask=1;
0401010   test   edx, eax        // if( (data & mask) !=0
0401012   je     401019h
0401014   test   cl, cl; sete cl // odd = ! odd;
0401019   add    eax,eax         // mask *=2
040101B   cmp    eax,100h        // mask <= 0x100
0401020   jle    401010h
}
00401022  movzx  eax,cl          //return odd;
/*...*/
00401026  ret
```

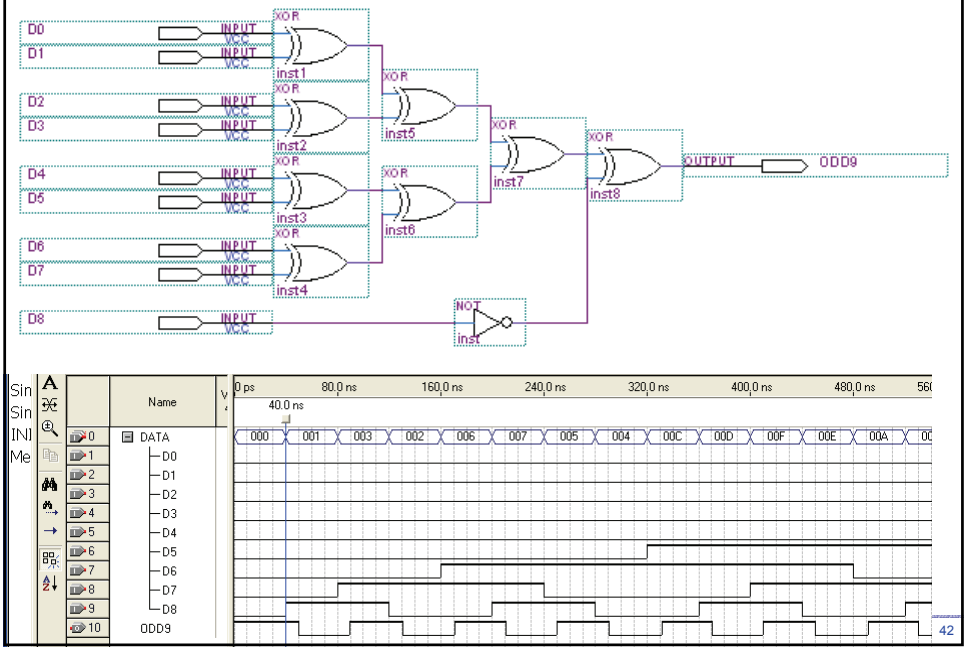
test ≡ logic and
sete cl ≡ zero-flag->cl

Conversion to parallel odd parity tester



41

Odd parity tester - optimized???



42

XOR – důležité vlastnosti

$x \oplus 0 = x$

$x \oplus 1 = \bar{x}$

často využíváno v logických obvodech pro řízení polarity signálu

$x \oplus x = 0$ *časté v procesorech pro nulování registru*

$x \oplus \bar{x} = 1$

$x \oplus \bar{y} = \bar{x} \oplus y = \overline{(x \oplus y)}$

XOR je komutativní a asociativní operace.

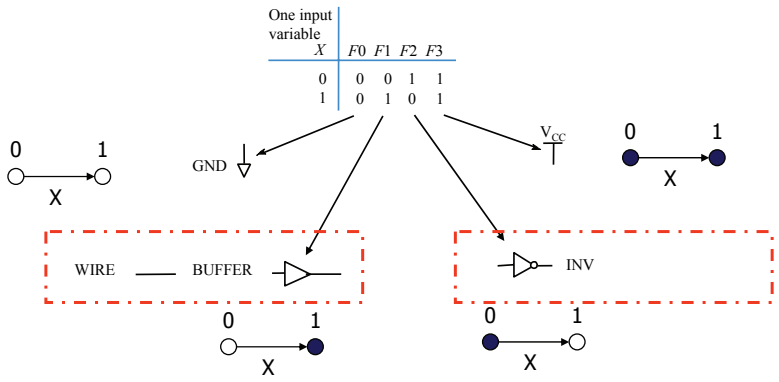
$x \oplus y = y \oplus x$

$(x \oplus y) \oplus z = x \oplus (y \oplus z)$

Logická funkce

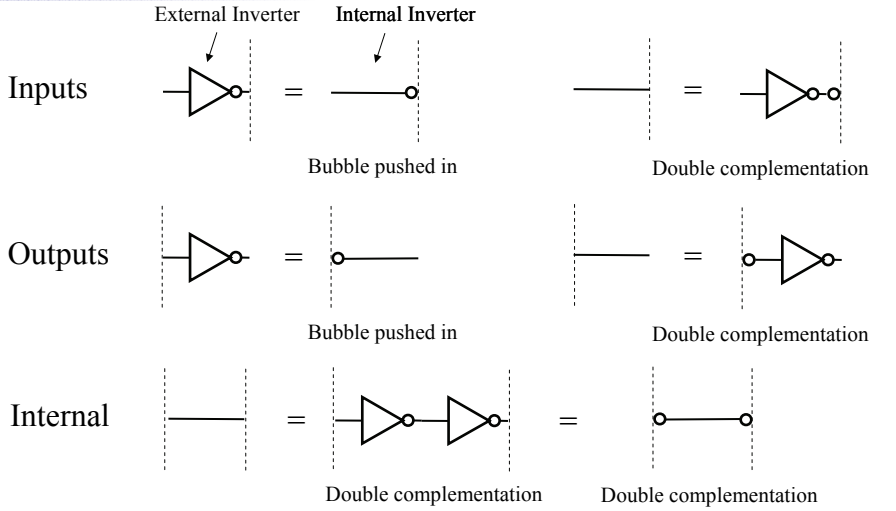
realizace

Všechny funkce 1 vstupních proměnné



Podle Richard S. Sandige, Digital Design Essentials, Prentice-Hall, 2002. p 130

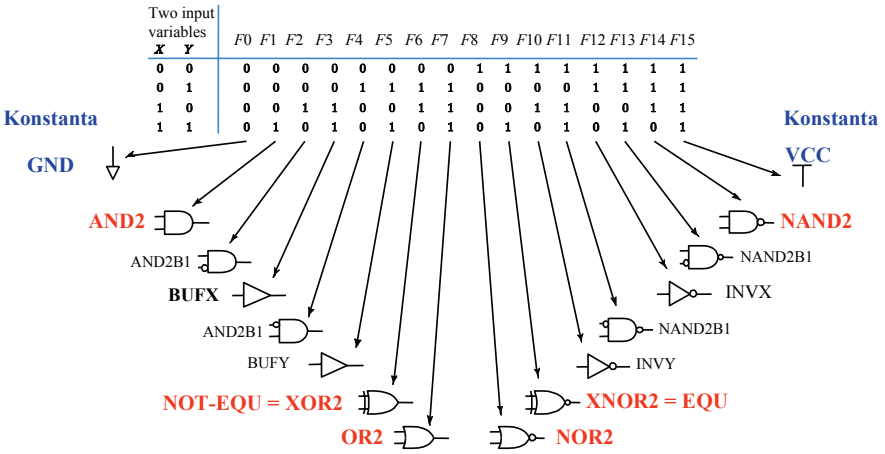
Negace na signálových vodičích



Richard S. Sandige, Digital Design Essentials, Prentice-Hall, 2002. p 141

Můžeme kreslit obvody v mnoha formách, graficky různých, ale funkčně ekvivalentních

Všechny funkce 2 vstupních proměnných

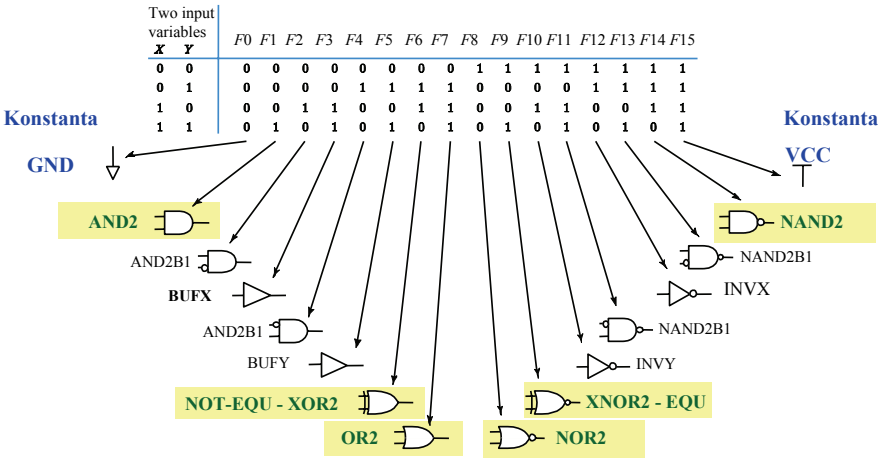


Which is the minimal number of logical connectives?

Functionally complete systems:

- One element
  - { nand2 } Scheffer's NAND
  - { nor2 } Pierce's NOR
- Two elements
  - { inv, and2 },
  - { inv, or2 },
  - and another 16 pairs

Všechny funkce 2 vstupních proměnných



Pouze zvýrazněné funkce se vyráběly jako IO

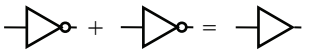
Boolean Functions of 2 Variables

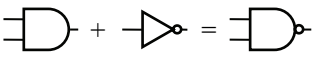
Function	Name	Comments
$F_0 = 0$	Null	Binary constant 0
$F_1 = x \cdot y$	AND	x and y
$F_2 = x \cdot y'$	Inhibition	x, but not y
$F_3 = x$	Transfer	x
$F_4 = x' \cdot y$	Inhibition	y, but not x
$F_5 = y$	Transfer	y
$F_6 = x \cdot y' + x' \cdot y$	Exclusive OR	x or y, but not both
$F_7 = x + y$	OR	x or y (or both)
$F_8 = (x+y)'$	NOR	NOT-OR
$F_9 = x \cdot y + x' \cdot y'$	Equivalence	x equals y
$F_{10} = y'$	Complement	NOT y
$F_{11} = x + y'$	Implication	If y, then x
$F_{12} = x'$	Complement	NOT x
$F_{13} = x' + y$	Implication	If x, then y
$F_{14} = (x \cdot y)'$	NAND	NOT-AND
$F_{16} = 1$	Identity	Binary constant 1

Standardní hradla

Buffer	$Y = a$	Invertor	$Y = \overline{a}$
AND	$Y = a\,b$	NAND	$Y = \overline{a\,b}$
OR	$Y = a + b$	NOR	$Y = \overline{a + b}$
XOR	$Y = a\overline{b} + \overline{a}b$ $Y = a \oplus b$	XNOR	$Y = ab + \overline{a}\,\overline{b}$ $Y = a \equiv b$

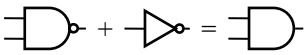
Negace



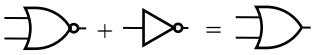


AND + NOT = NAND

OR + NOT = NOR



NAND + NOT = AND



NOR + NOT = OR

Různé znaky pro operace Booleovské algebry

Název	NOT	AND	OR	XOR
výstup je 1 když:	Vstup 0	Oba vstupy jsou 1	Aspoň jeden vstup 1	Vstupy jsou různé
Operátor a některé alternativy	x' $\neg x$ or $\overline{x}$	$x \,.\, y$ $x \wedge y$	$x + y$ $x \vee y$	$x \oplus y$ $x \not\equiv y$
Možný zápis ve výrazu	$-x$	$x \times y$ či $x.y$	$x + y$	x xor y
Bitové operace C,C#, Java	~x	x & y	x | y	x ^ y
Schémata symbol				

V textu budeme používat zvýrazněné symboly

Variations in Gate Symbols

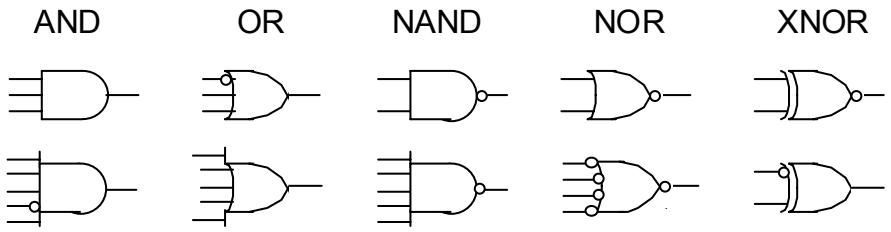
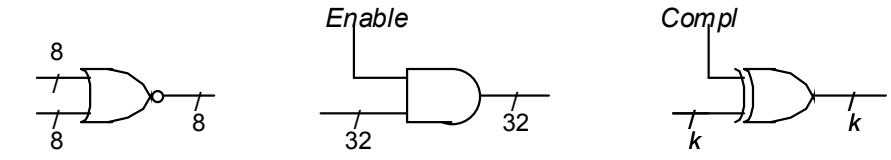


Figure 1.2 Gates with more than two inputs and/or with inverted signals at input or output.

Signal Bundles



(a) 8 NOR gates (b) 32 AND gates (c) *k* XOR gates

Figure 1.5 Arrays of logic gates represented by a single gate symbol.

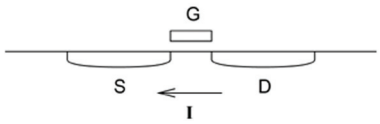
Logická funkce

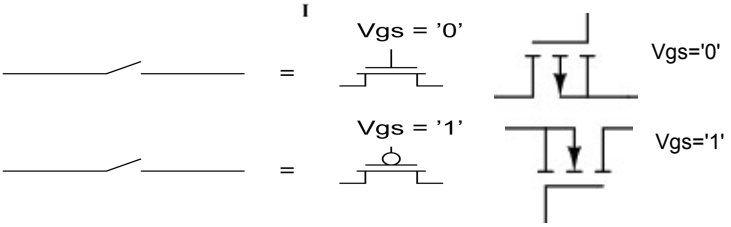
realizace

CMOS Devices

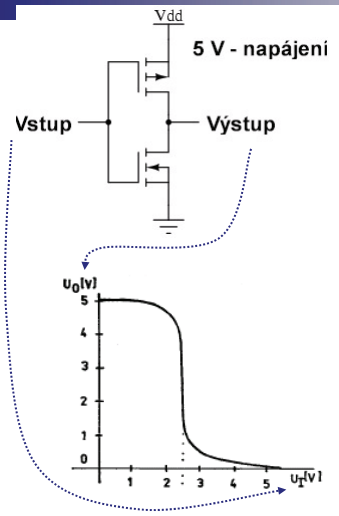
Complementary MOSFET (Metal Oxide Semiconductor Field Effect Transistor).

Cross Section

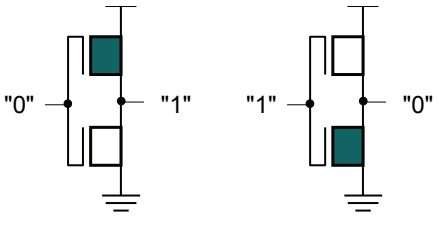


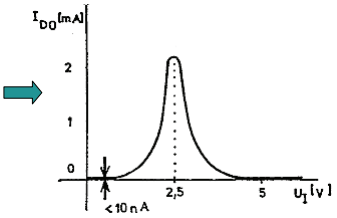


CMOS invertor



Static characteristic

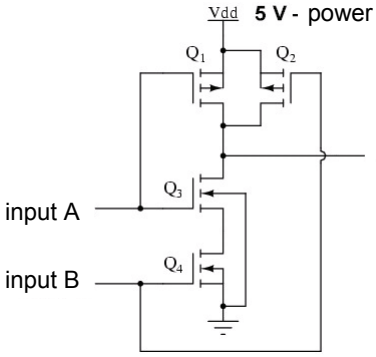


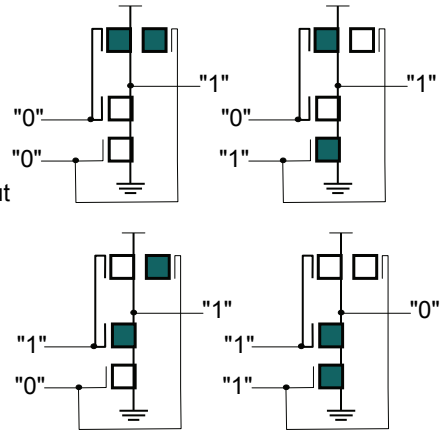


Power consuption

Even if circuits are logically inactive they are influence power source...

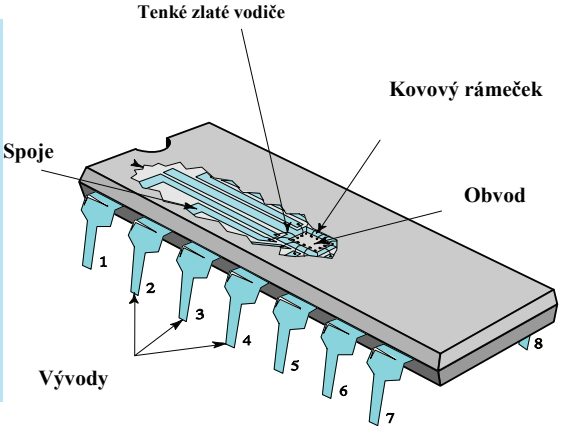
CMOS NAND gate

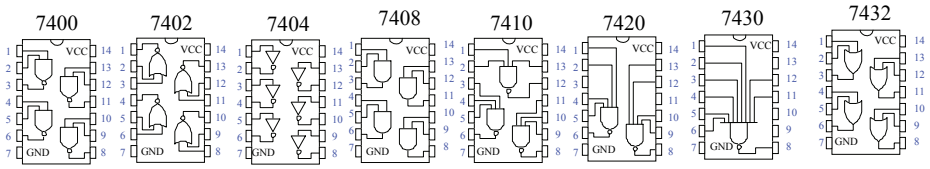




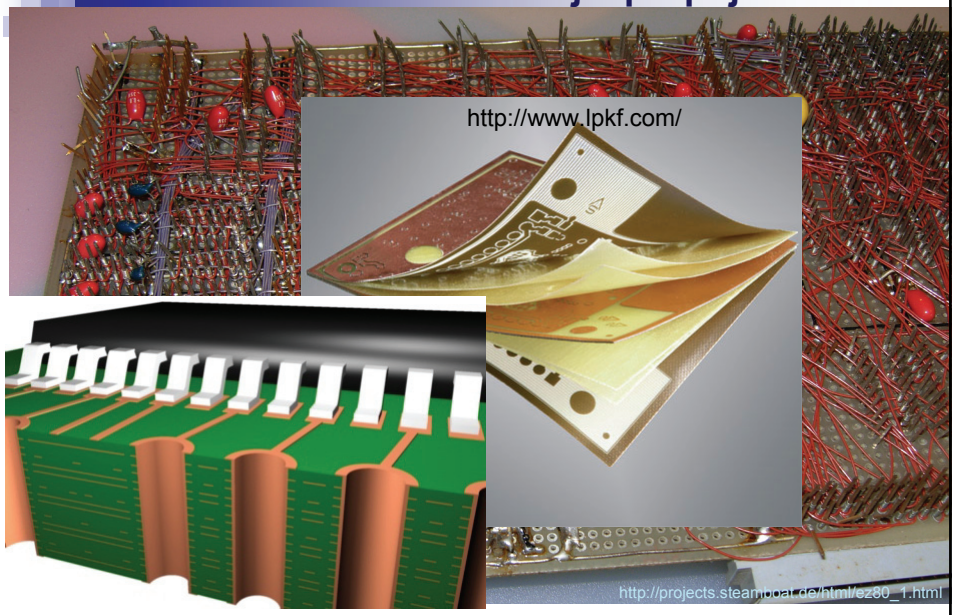
Hradla logické obvody řady 7400

TTL IO 74… se dnes již skoro nepoužívají, ale jejich řada od 7400 cca do 74899 zůstává vhodnou typovou kolekcí obvodů pro návrhy.
Seznam viz. Wikipedia („seznam obvodů řady 74“)



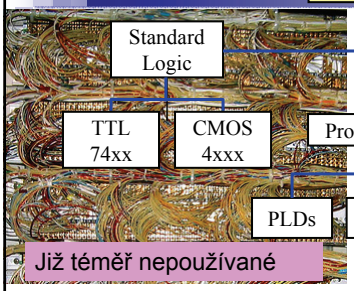


Drahé vnější propojení obvodů



http://www.elkosoft.com/md/mdhelp/chapters/triplate.html

Moderní způsoby realizace logiky



Standard Logic

TTL 74xx

CMOS 4xxx

PLDs

Již téměř nepoužívané

ASICs

Programmable

Semi-Programmable

Full Custom

CPLDs

FPGAs

Gate Arrays

Standard Cells

Microprocessor & RAM

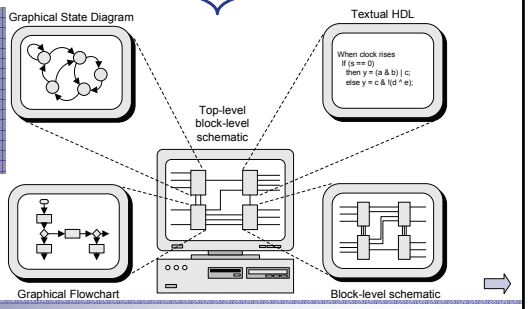
>=1 >2000 >4000 >cca 20 tisíc až 10 milionů

rentabilita

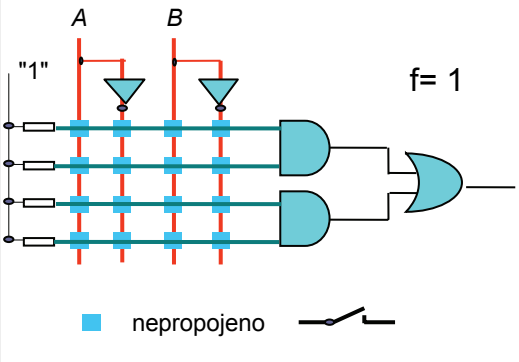
ASIC-Application Specific Integrated Circuits

Programovatelné

- Programmable Logic Device (**PLD**)
- Complex PLD (**CPLD**)
- Field Programmable Gate Arrays (**FPGA**)



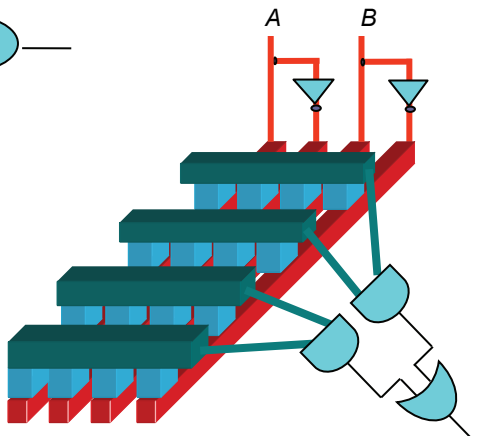
Princip PLD 1/4



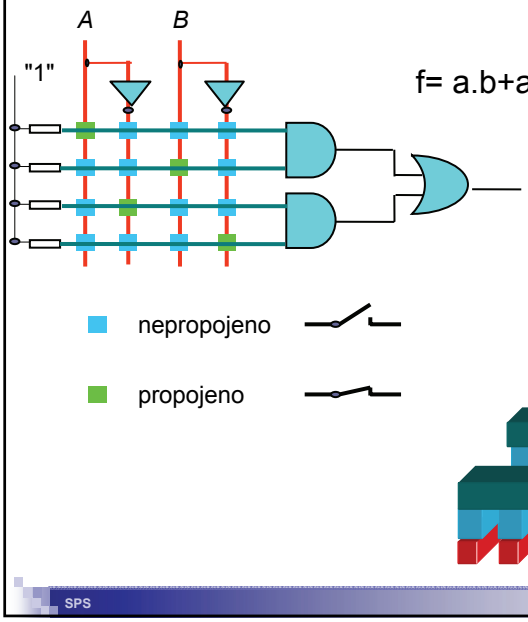
f= 1

nepropojeno

3D

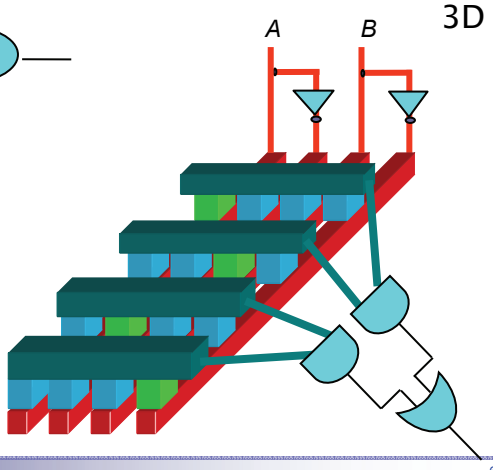


Princip PLD 2/4

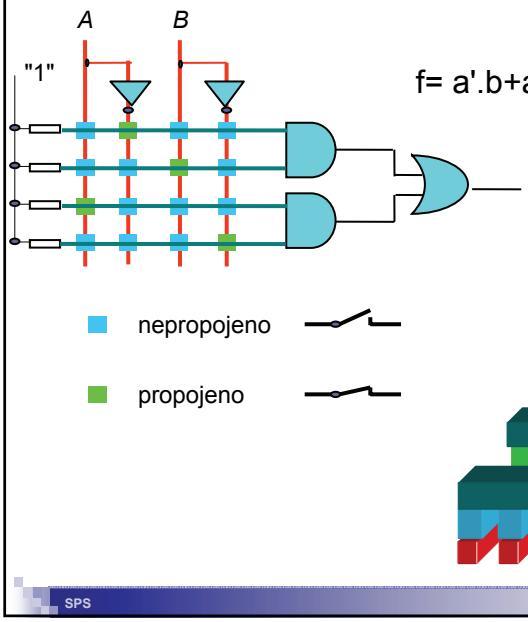


f= a.b+a'.b' = a ≡ b

3D

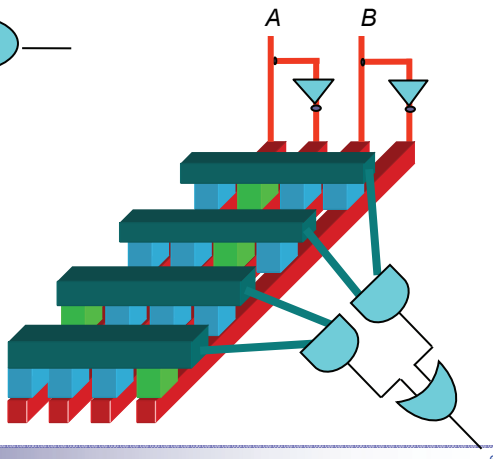


Princip PLD 3/4

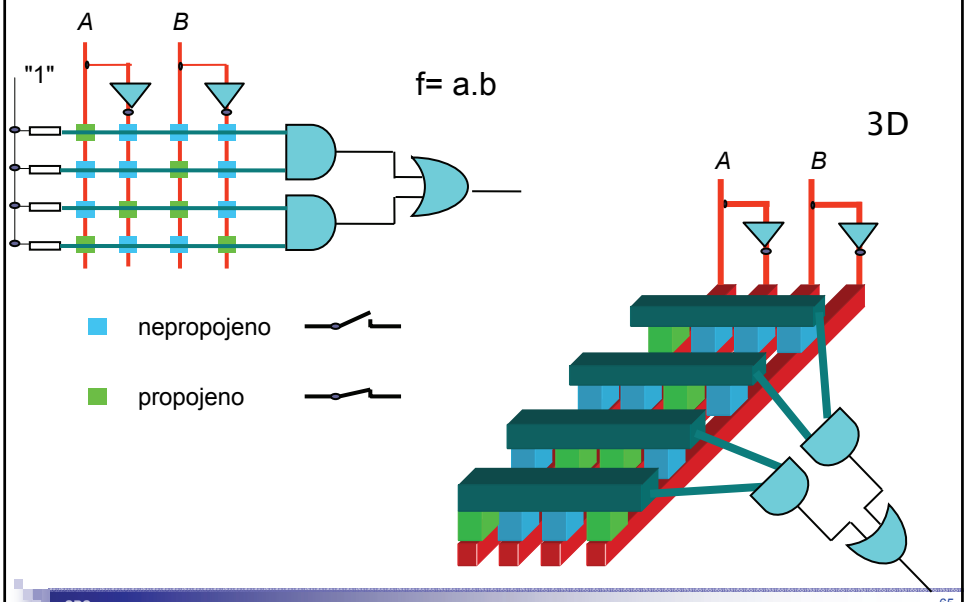


f= a'.b+a.b' = a xor b

3D

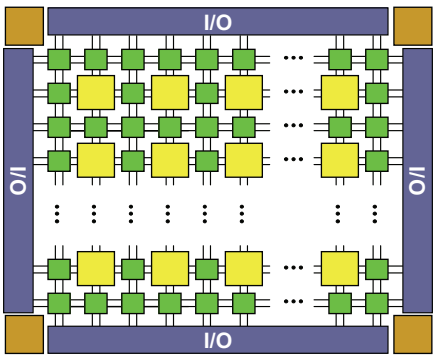


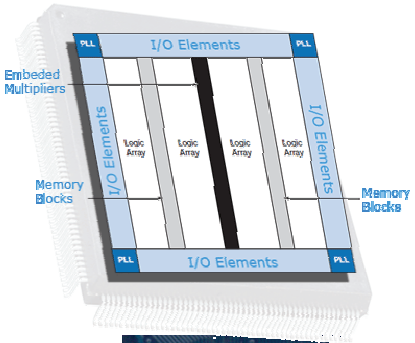
Princip PLD 4/4



Běžná architektura FPGA

Logická či speciální funkce Propojovací přepínače

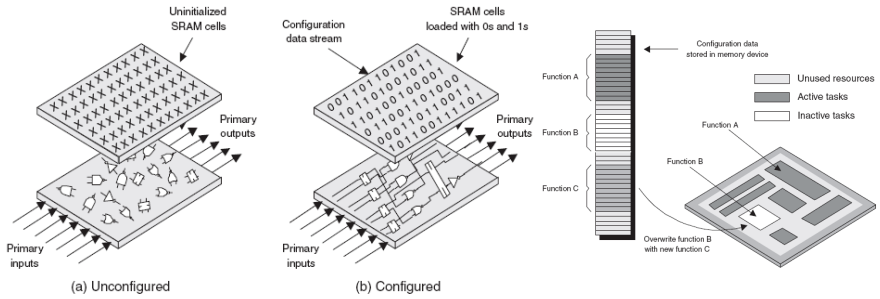




Programovatelné I/O bloky

Distribuce hodin, fázové závěsy (*PLL – Phase-locked-loop*)

Konfigurace FPGAs



Zdroj: S. Reda, FPGA

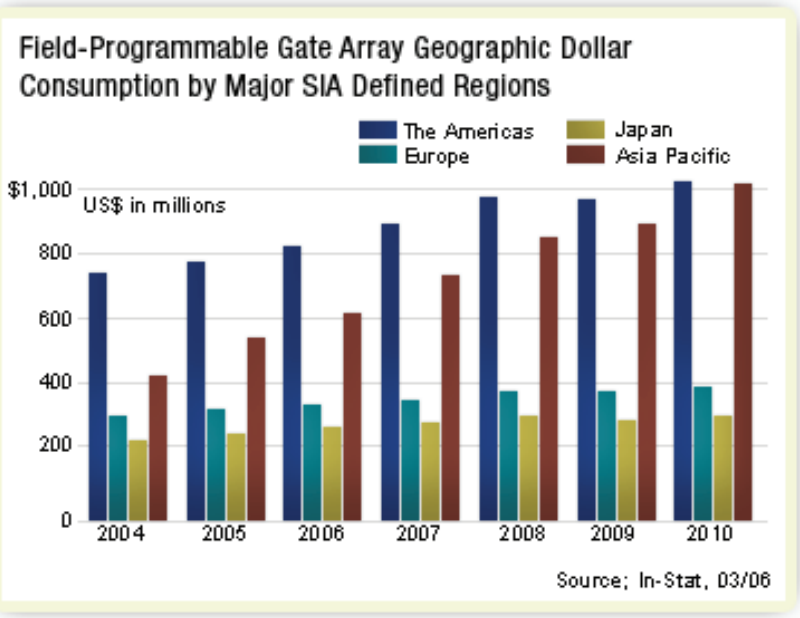
FPGA pozice

Lehké programování		Obtížné programování
Flexibilita		Pevná funkce
Nízký výkon pro některé operace		Vysoký výkon pro zamýšlené operace

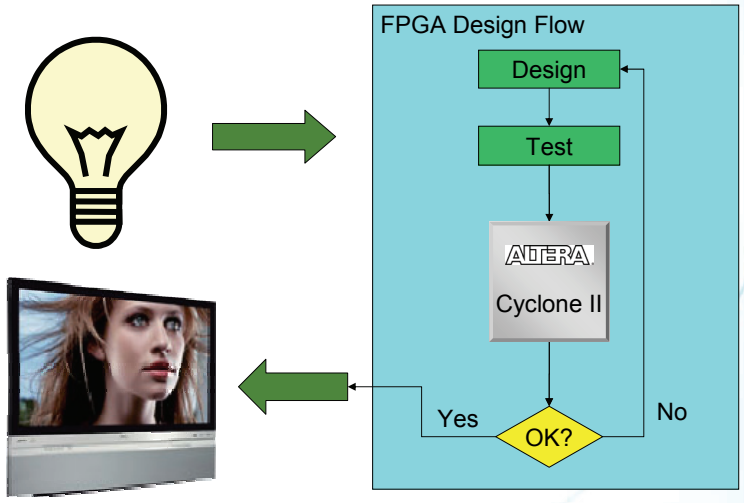
Full custom Micro-processor | Full custom DSP | FPGA | Semi-prog. ASIC

Výrobci nabízejí širokou škálu FPGA obvodů

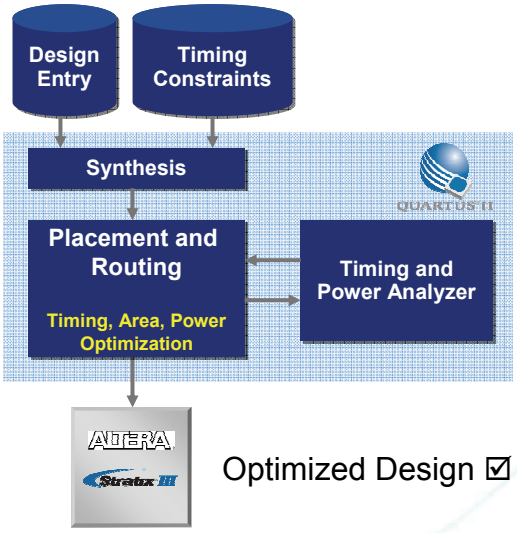
DSP = Digital signal processor



Getting a Product Out



Quartus II Design Flow

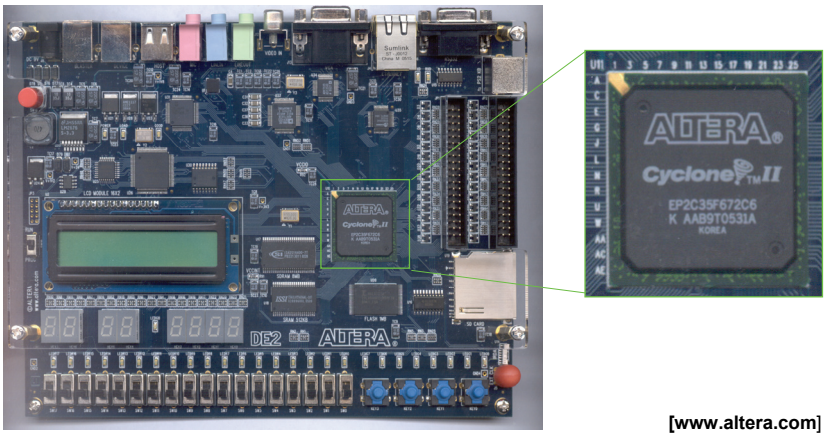


Optimized Design ☑

ALTERA.

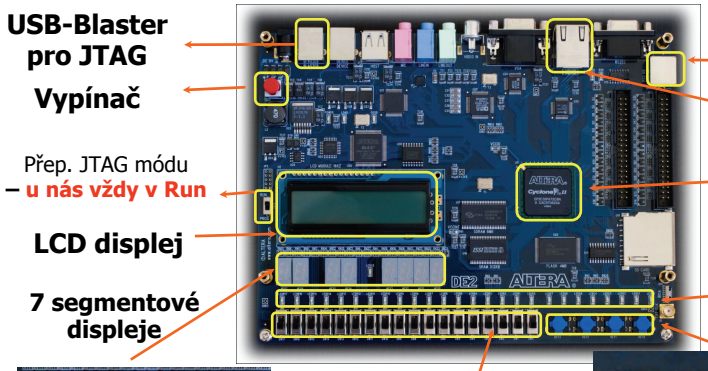
FPGA Example

- Altera Development Education Kit DE2

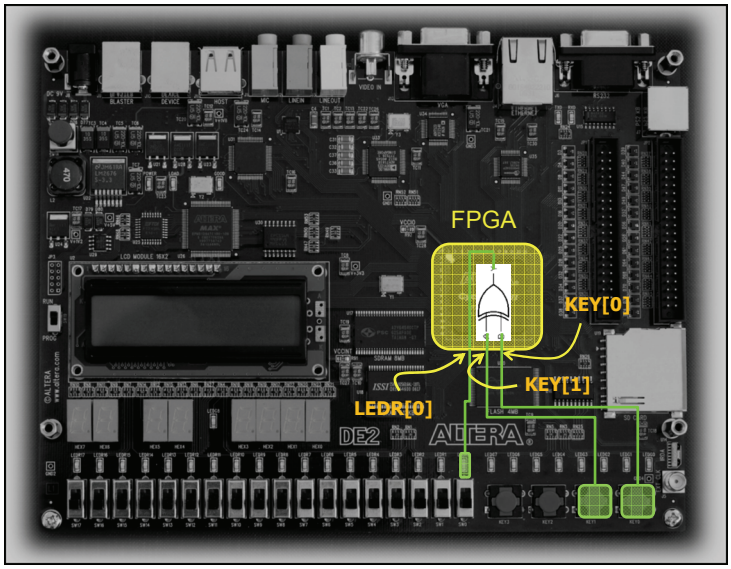


[www.altera.com]

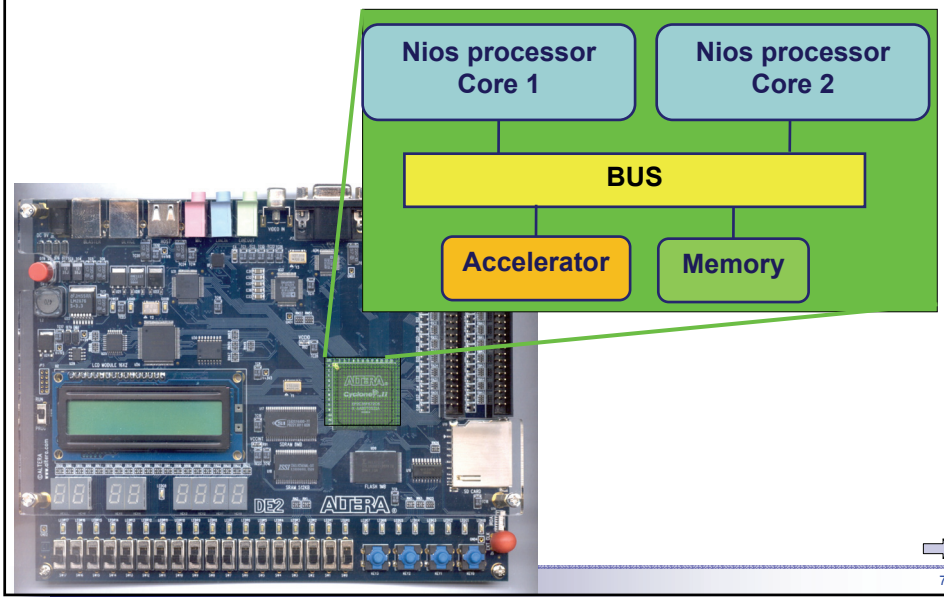
Vybrané prvky DE2



Příklad jednoduchého obvodu



Příklad návrhu: Víceprocesorový systém



Příklady systémů na DE2