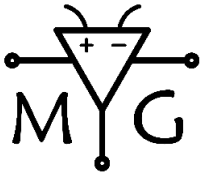


bastlíři SH

MacGyver

macgyver.siliconhill.cz

Když procesor nestačí, FPGA zaskočí

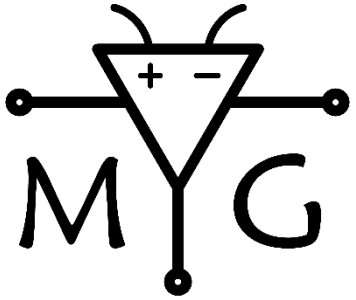
Jan „Fosfor“ Pospíšil

8. 12. 2015

Středisko UN*Xových technologií

Úterní díl Bastlířských Střed

Bastlířské středy



- (Ne)pravidelné semináře projektu *MacGyver – bastlíři SH*
- http://macgyver.siliconhill.cz/wiki/stredy
- Bylo
  - 18. 11. – Arduino a debugger
  - 25. 11. – Technologie návrhu a výroby DPS
  - 2. 12. – O napájecím (sub)systému
- Bude
  - 16. 12. – Stručná historie zpětné vazby

O ~~mě~~ nás

- Jan „Fosfor“ Pospíšil
  - FEL od 2005, SH od 2006, FIT od 2011, CERN od 2015
  - *MacGyver – bastlíři SH*, elektronika, FPGA, vesmír
  - fosfor@sh.cvut.cz
  - http://linkedin.com/in/fosfor
- Vojta Suk
  - FEL od 2009, dnes Java EE
  - *MacGyver – bastlíři SH*, elektronika
  - v.suk@sh.cvut.cz
  - http://twitter.com/VojtechSuk

Co nás čeká

- Úvod do logiky (příklad poprvé)
- Poznej nepřítele
- O FPGA
- FPGA design flow (příklad podruhé)
- Vnitřnosti FPGA (příklad znovu a lépěji)
- Závěrečné slovo

- Volitelně – rozšířená ukázka

Úvod do logiky

- Matematická logika
- Dvouprvková Booleova algebra
  - 0/1, high/low, true/false, (ne)pravda
- Vystačíme si s dvěma hodnotami

Binární kódování
000
001
010
011
100
101
110
111

Kombinační logika

- Logické funkce (i více vstupů), např.:

a	NOT a
0	1
1	0

a	b	a AND b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a OR b
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

- Výsledek funkce závisí jen na jejích vstupech

$$\vec{y} = f(\vec{x})$$

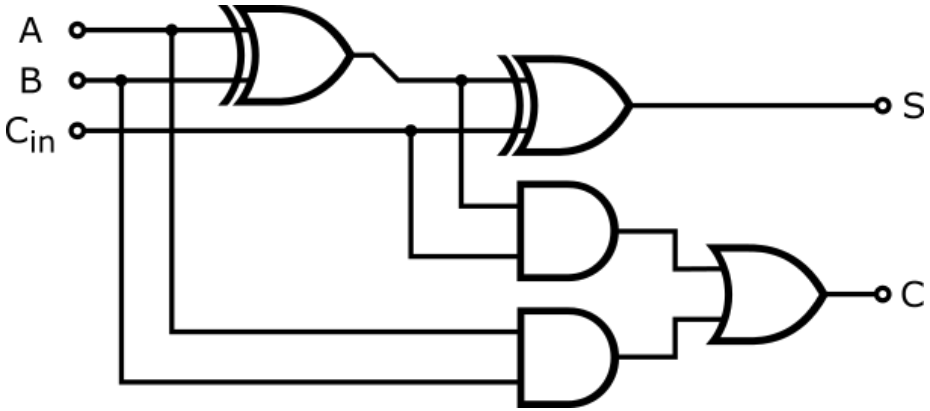
- Pouze zpoždění dané technologií (zpoždění hradel)

Hradlo – základní realizace logické funkce

Příklad: návrh sčítačky

- Bitová sčítačka: S = A + B
  - Vstup: 2 x 1 bit (operandy) + 1 bit (přenos z nižšího řádu)
  - Výstup: 1 bit (součet) + 1 bit (přenos do vyššího řádu)

$$C = (A \wedge B) \vee (Cin \wedge (A \oplus B))$$
$$S = A \oplus B \oplus Cin$$



wikipedia.org

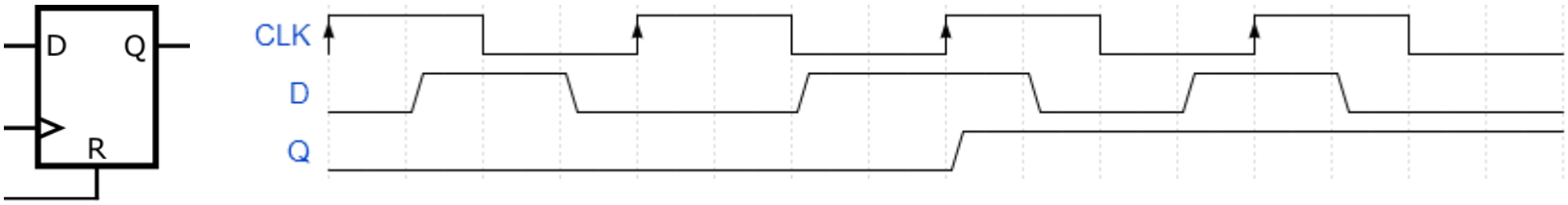
A	B	Cin	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sekvenční logika

- Výsledek závisí na vstupech a minulém stavu

$$\overrightarrow{y_2} = f(\overrightarrow{x_2}, \overrightarrow{y_1})$$

- Paměť – např. klopný obvod D (a další)
- Časováno (taktováno) hodinami
  - Takt je vymezen vzestupnou hranou (nebo sestupnou, úrovní)
  - Omezeno shora maximálním zpožděním hradel

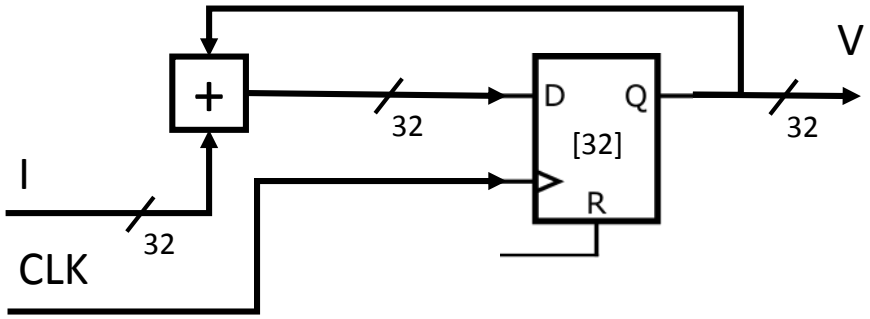


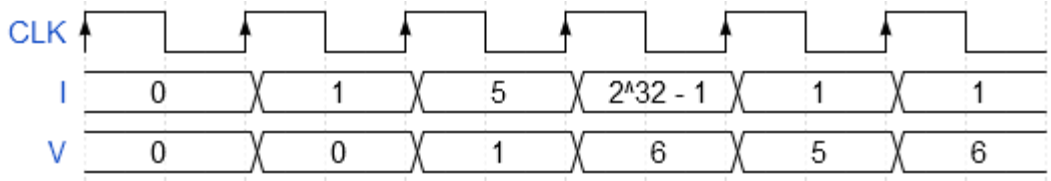
wikipedia.org

Příklad: návrh čítače

- 32 bitový čítač (modulo 2^{32}) s volitelným krokem
  - Vstup: hodiny (CLK), krok (I)
  - Výstup: načítaná hodnota (V)

$$V_x = (V_{x-1} + I) \; mod \; 2^{32}$$



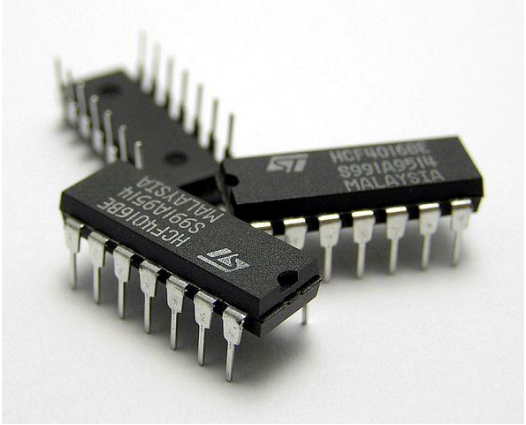


Poznej nepřítele

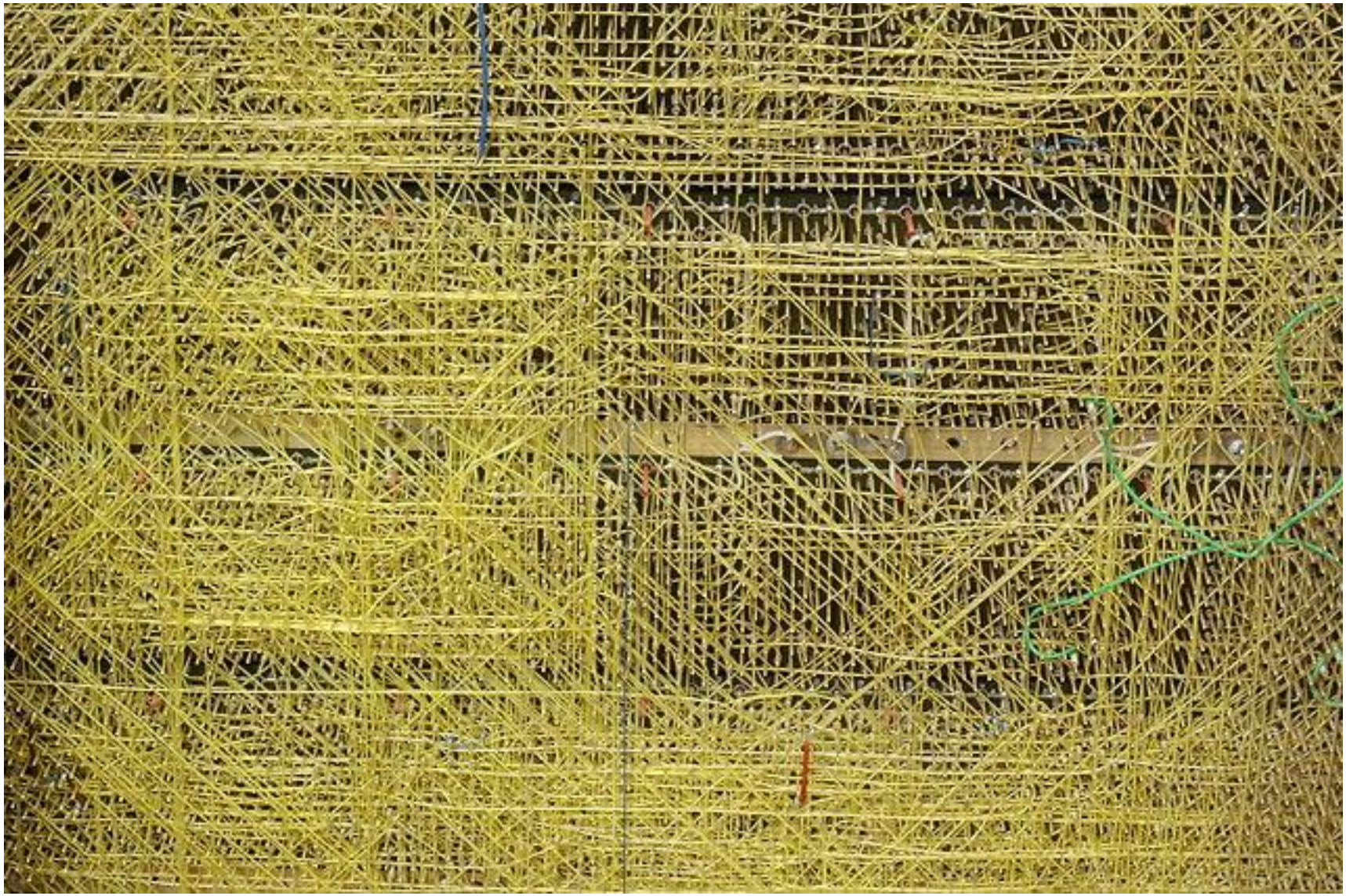
- Diskrétní integrované obvody a plno drátů
- Zákaznický integrovaný obvod na vlastním křemíku
- (Mikro)procesor a „normální“ programování

Diskrétní logické součástky

- Jednotky hradel v pouzdru
- Pro malá zapojení dokonalý přehled, snadná manipulace, nízká cena
- Vše se však rychle zhoršuje s rostoucí velikostí zapojení…



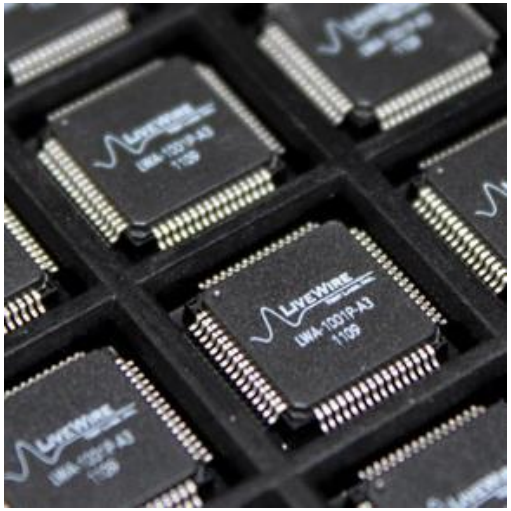
wikipedia.org



wikipedia.org

Zákaznický integrovaný obvod

- ASIC (Application-specific integrated circuit)
- Nízká cena (výsledného obvodu) při masové výrobě
- Dlouhý a drahý vývoj
  - Měsíce a více
  - I několik milionů $



wikipedia.org

(Mikro)procesor

- Hardware „jen“ použijeme, tvoříme software
- Univerzálnost, nízká cena, hodně rozšířené
- Platí se rychlostí
  - u složitějších operací
  - u přenosů většího množství dat

```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author john doe <doe.j@example.com>
 */
HelloButton()
{
  JButton hello = new JButton( "Hello, wor
  hello.addActionListener( new HelloBtnList

  // use the JFrame type until support for t
  // new component is finished
  JFrame frame = new JFrame( "Hello Button"
  Container pane = frame.getContentPane();
  pane.add( hello );
  frame.pack();
  frame.show();           // display the fra
}
```

O FPGA

- Field-programmable gate array
- „F-terénu Programovatelné Hradlové Pole“

- Pole – je tam toho hodně
- Hradlové – je tam hodně hradel
- Programovatelné – ta hradla jdou programovat
- F-terénu – a jde to programovat kdykoliv, dokonce i za běhu

Programovatelná logika

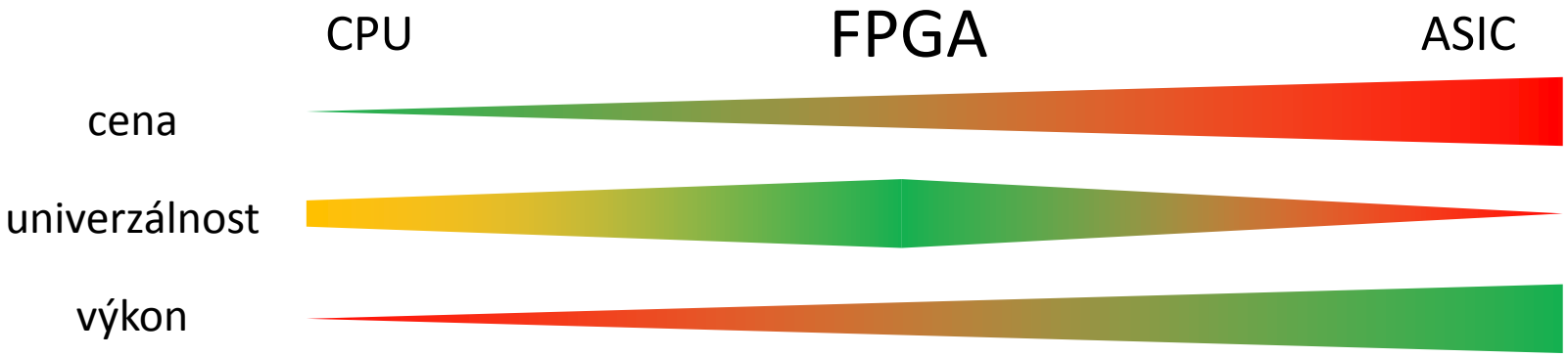
- Virtuální hromada 7400 obvodů a plno drátů
- Virtuální vývoj ASIC
- „Programovaní“ – popis zapojení
- FPGA jsou nejmodernější součástky v této oblasti
  - Historie: (C)PLD, GAL, PAL, (E)EPROM, …

Na co použijeme FPGA

- Prototypování, vývoj ASIC
- Rychlé zpracování dat, potřeba nízké latence
- Velké množství vstupů/výstupů, dat
- Na to, co by na (M)CPU trvalo příliš dlouho a na co nemáme ASIC

Porovnání s nepřáteli

- Není to ASIC, ale je to skoro stejně rychlé
  - Je to logický integrovaný obvod
- Není to (M)CPU, ale je to programovatelné
  - Je to programovatelný logický obvod
- Vyrábí se to masově – je to levné
  - Může být levnější než ASIC, ale dražší než (M)CPU



Výrobci, jejich IDE a (budoucí) největší FPGA

- Xilinx, http://www.xilinx.com/
  - IDE: Vivado, dříve ISE
  - Virtex UltraScale+: 16 nm, 3,5 milionu „logických buněk“, IO až 33 Gb/s, 832 IO pinů, DSP 6,3 TMAC/s
- Altera, https://www.altera.com/
  - IDE: Quartus II
  - Stratix X: 14 nm, 5,5 milionu „logických elementů“, IO až 30 Gb/s, 1640 IO pinů, DSP 7,9 TMAC/s

Výrobci, jejich IDE a vlajková FPGA

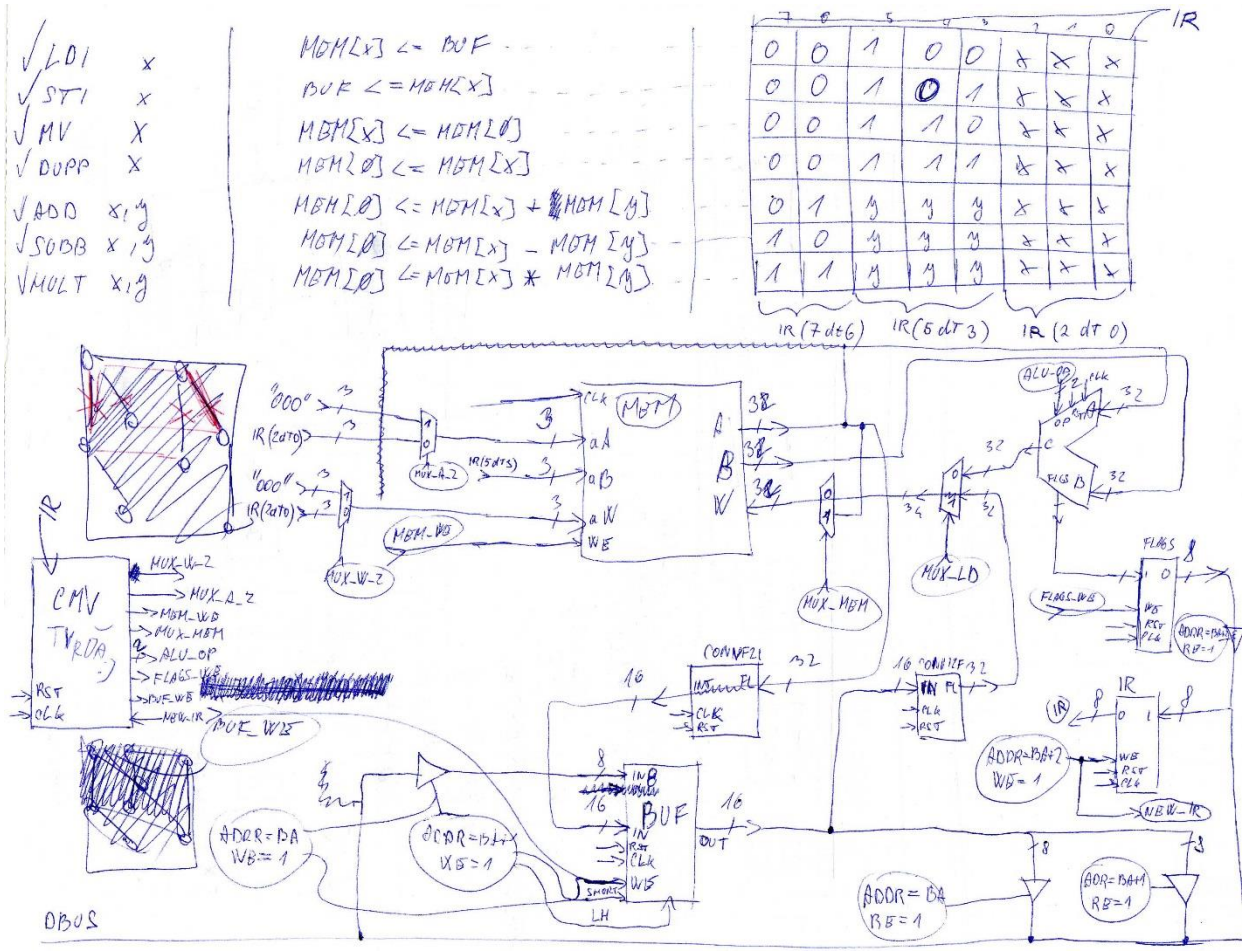
- Lattice, http://www.latticesemi.com/
  - IDE: Diamond
  - Malá FPGA s nízkým odběrem, levná
- Microsemi, http://www.microsemi.com/
  - IDE: Libero
  - Radiačně odolná FPGA

FPGA design flow

- … aneb cesta od myšlenky, k blikající LEDce
- „Frontend“ spíše univerzální, občas i opensource
- „Backend“ spíše uzavřený, proprietární
  - Výrobci FPGA nesdělí podrobnosti o svých zařízeních
  - Pro finální kroky je nutné používat nástroje výrobce

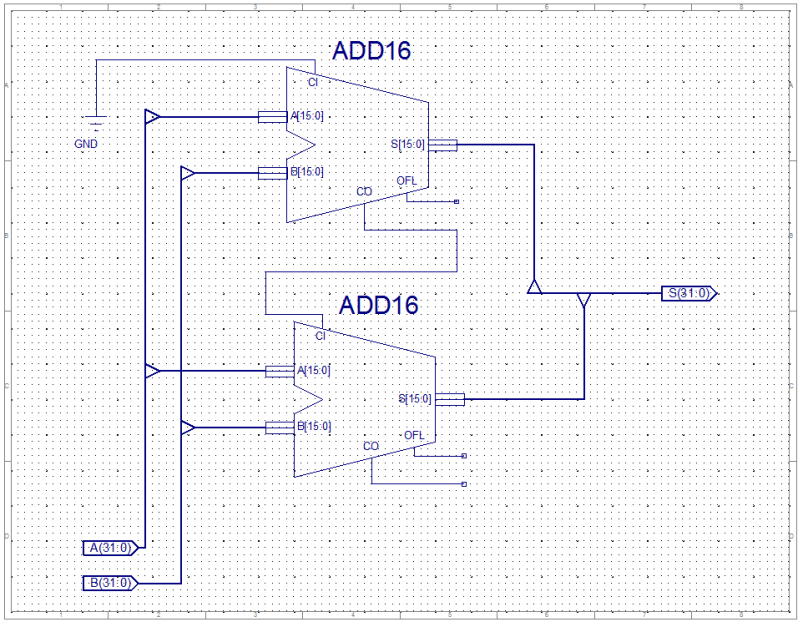
Příklad FPGA design flow

1. Myšlenka, návrh



Příklad základního design flow

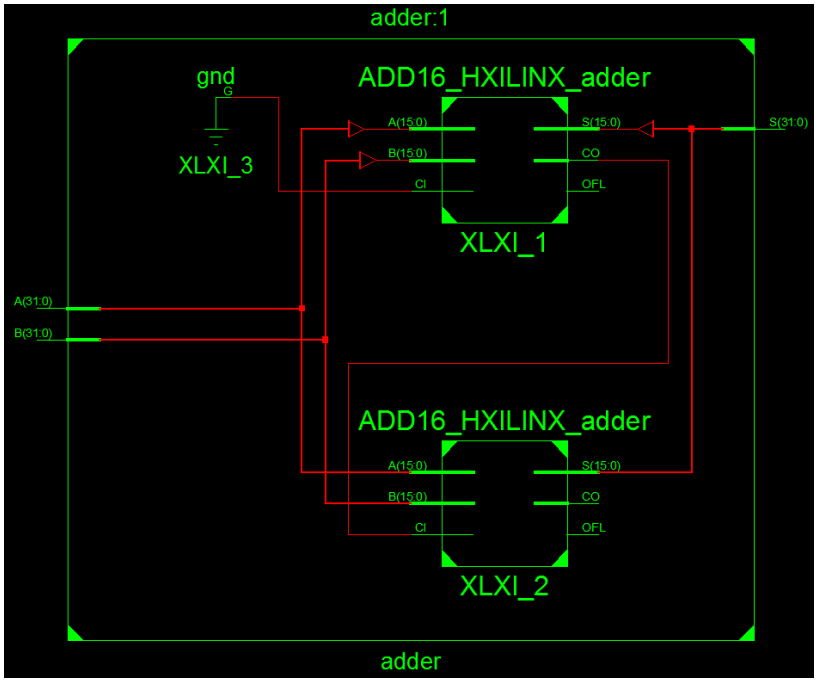
1. Myšlenka, návrh
2. Schéma / kód (VHDL, (System)Verilog, SystemC,…)




```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity ADD16_HXILINX_adder is
port(
        CO   : out std_logic;
        OFL  : out std_logic;
        S    : out std_logic_vector(15 downto 0);

        A    : in std_logic_vector(15 downto 0);
        B    : in std_logic_vector(15 downto 0);
        CI   : in std_logic
    );
end ADD16_HXILINX_adder;

architecture ADD16_HXILINX_adder_V of ADD16_HXILINX_adder is
  signal adder_tmp: std_logic_vector(16 downto 0);
begin
   adder_tmp <= conv_std_logic_vector(
     (conv_integer(A) + conv_integer(B) + conv_integer(CI)),17);
   S          <= adder_tmp(15 downto 0);
   CO         <= adder_tmp(16);
   OFL <=  ( A(15) and B(15) and (not adder_tmp(15)) ) or
           ( (not A(15)) and (not B(15)) and adder_tmp(15) );

end ADD16_HXILINX_adder_V;
```

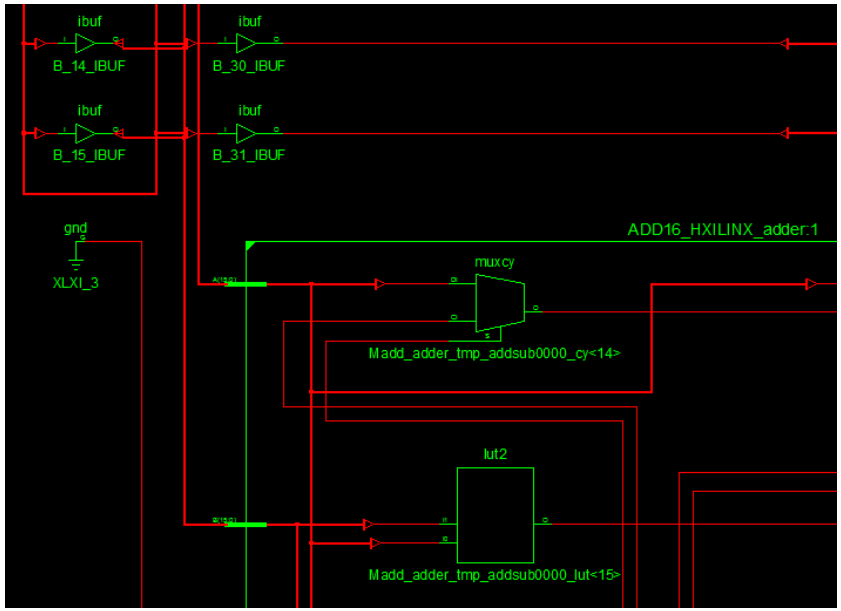
Příklad základního design flow

1. Myšlenka, návrh
2. Schéma / kód (VHDL, (System)Verilog, SystemC,…)
3. Syntéza → netlist (aneb opět schéma)



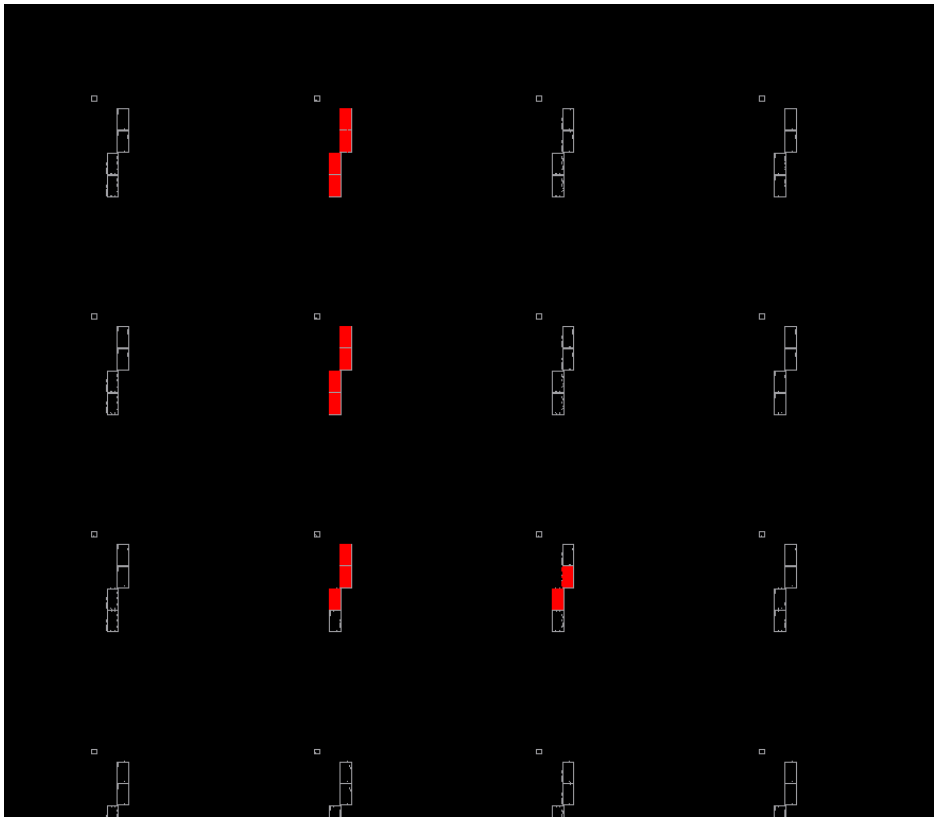
Příklad základního design flow

1. Myšlenka, návrh
2. Schéma / kód (VHDL, (System)Verilog, SystemC,…)
3. Syntéza → netlist (aneb opět schéma)
4. Mapování na technologie → netlist



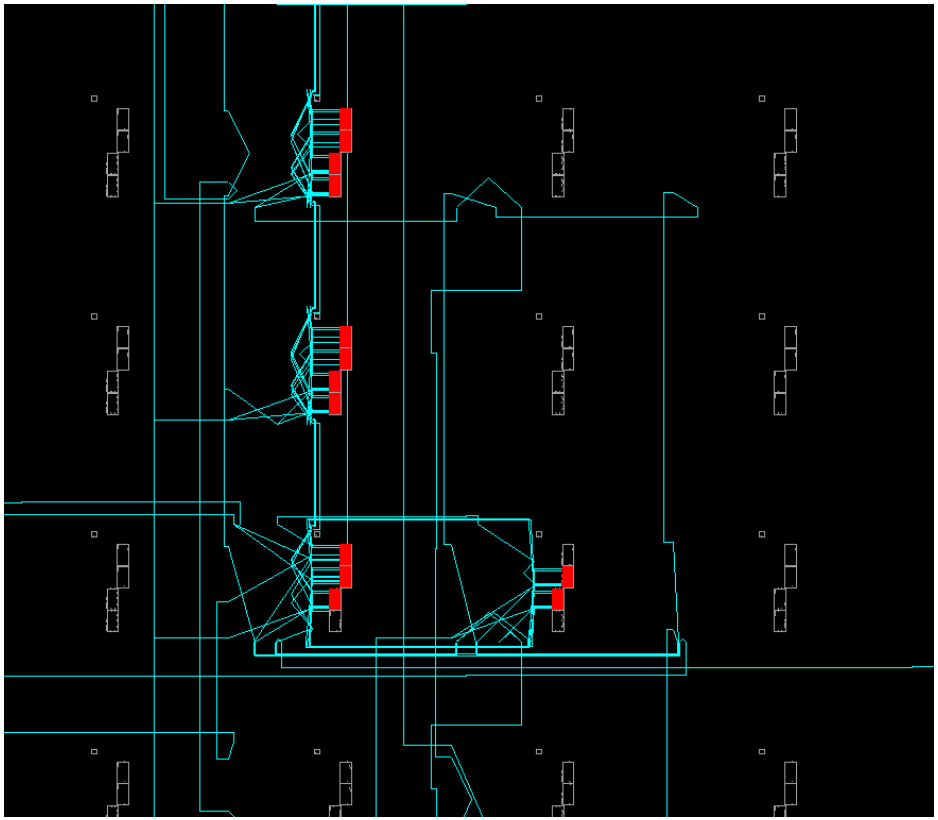
Příklad základního design flow

4. Mapování na technologie → netlist
5. Place (rozmístění) → víme co kde je



Příklad základního design flow

5. Place (rozmístění) → víme co kde je
6. Route (zapojení) → teď už víme i jak to je spojené



Příklad základního design flow

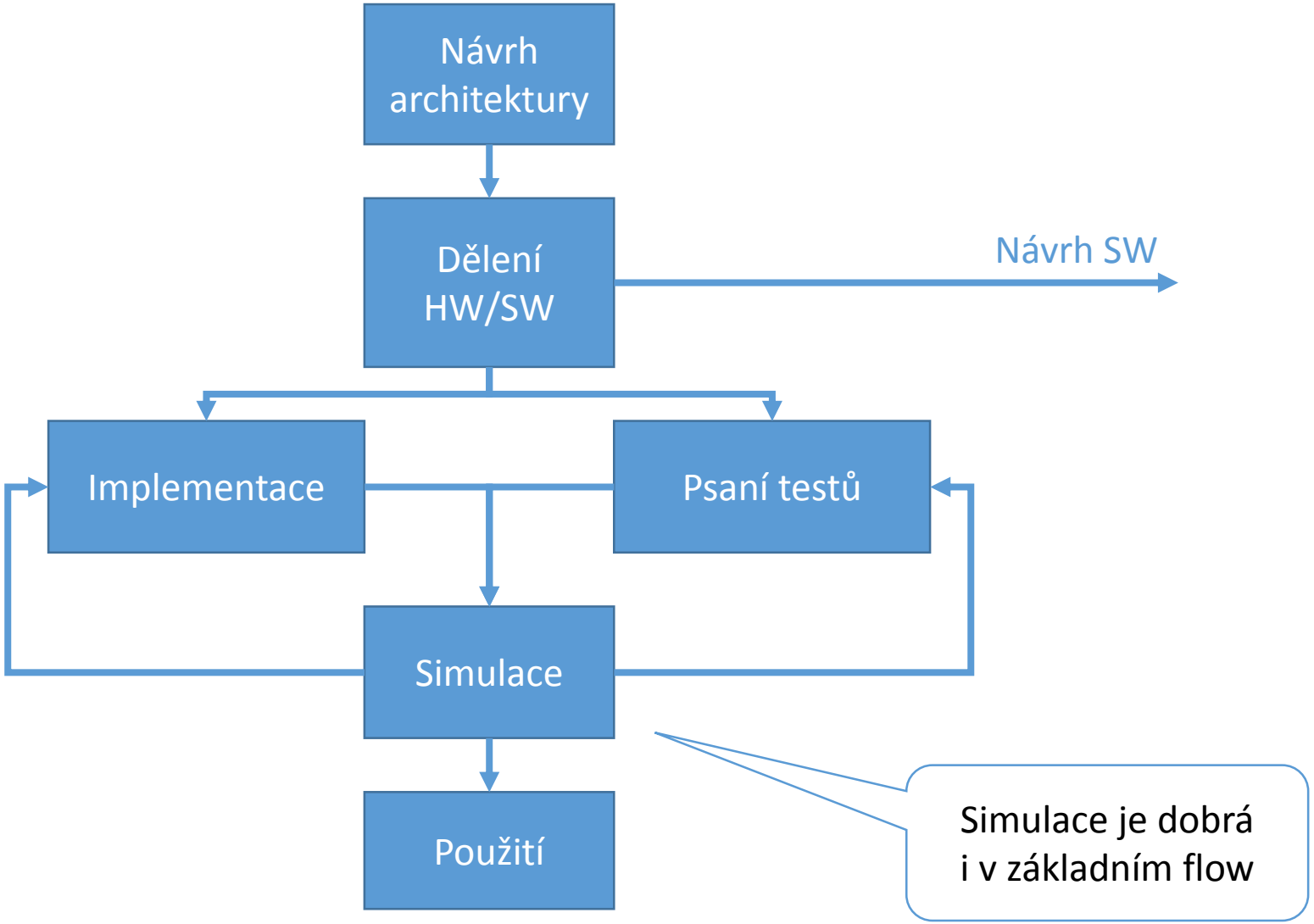
1. Myšlenka, návrh
2. Schéma / kód (VHDL, (System)Verilog, SystemC,…)
3. Syntéza → netlist (aneb opět schéma)
4. Mapování na technologie → netlist
5. Place (rozmístění) → víme co kde je
6. Route (zapojení) → teď už víme i jak to je spojené
7. Vytvoření bitstreamu (programovací data)

```
1  Xilinx ASCII Bitstream
2  Created by Bitstream P.20131013
3  Design name:     top.ncd;UserID=0xFFFFFFFF
4  Architecture:    spartan3a
5  Part:            3s700anfgg484
6  Date:            Sun Dec 06 22:13:43 2015
7  Bits:            2732640
8  1111111111111111
9  0011110000001111
10 0011000110100001
11 0000000011001001
```

Ukázka v Xilinx ISE

- Sčítačka schématem
- Implementace
- Simulace
- Ukázka na HW

Příklad rozšířeného design flow



Vnitřnosti FPGA

- Aneb jak ty trpaslíčci vlastně vypadají?

- Kombinační logika
- Sekvenční logika
- Vstupy / výstupy
- Propojení (všeho)
- Konfigurační paměť
- Bloky dalších funkcí

Vnitřnosti FPGA

- Aneb jak ty trpaslíčci vlastně vypadají?

• Kombinační logika	LUT (Look-up table) } „logický blok“
• Sekvenční logika	Dff (D flip-flop) } „logický blok“
• Vstupy / výstupy	IOB (in/output block)
• Propojení (všeho)	programmable interconnect
• Konfigurační paměť	CRAM
• Bloky dalších funkcí	JTAG, SPI, PLL, BRAM, DSP, CPU, …

Vše konfigurovatelné

LUT (Look-up table)

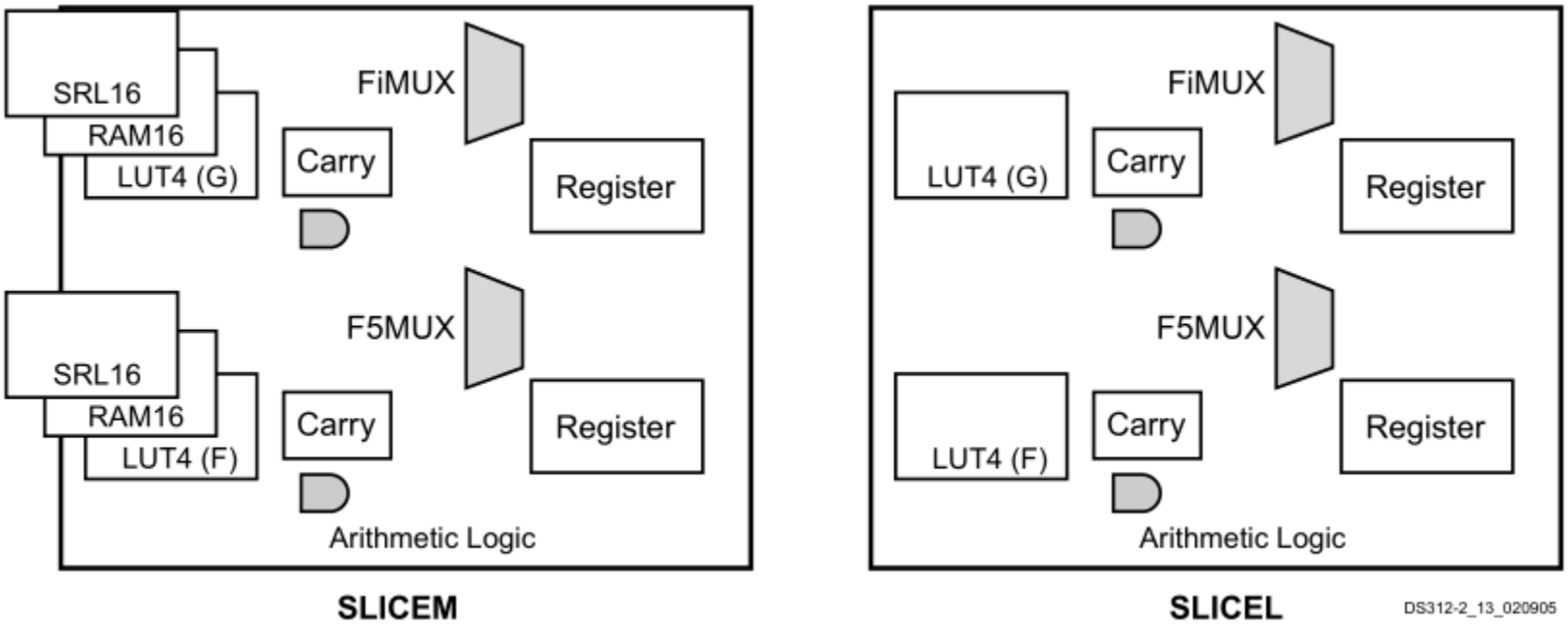
- Vyhledávací tabulka
- Logická funkce → pravdivostní tabulka → paměť
- Vstupy funkce = adresní bity do paměti
- Všechny výsledky funkce jsou předpočítané
- Např. 3 vstupový LUT
  - Libovolná logická funkce 3 proměnných
  - Paměť 8x1 bit = konfigurace 8 bitů
- Běžně 4-6 vstupový LUT

A	B	Cin	C ← LUT 1	S ← LUT 2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Logický blok

- Xilinx: CLB (Configurable Logic Block)
- Altera: LAB (Logic Array Block)
- Základní stavební blok opakující se struktury FPGA
- Obsahuje
  - Jednotky LUTů
  - Jednoty klopných obvodů D
    - Konfigurace několik bitů (reset stav, aktivní hrana, …) na KO
  - Trochu další logiky
  - Programovatelné propojení

Logický blok – Xilinx Spartan 3AN



Podrobněji: http://www.xilinx.com/support/documentation/user_guides/ug331.pdf#page=204

IOB (in/output block)

- Rozhraní s vnějším světem – pin
- Vstup / výstup / vstupovýstup
- Umí stav vysoké impedance
- Často obsahuje také klopný obvod
  - Zmenšení zpoždění na vstupu / výstupu
- Napěťové standardy, různá síla budiče
- Může obsahovat i DDR registry, zpožďovací linky, (de)serializátory, …
- I desítky bitů konfigurace

Pro Spartn 3AN: http://www.xilinx.com/support/documentation/user_guides/ug331.pdf#page=315

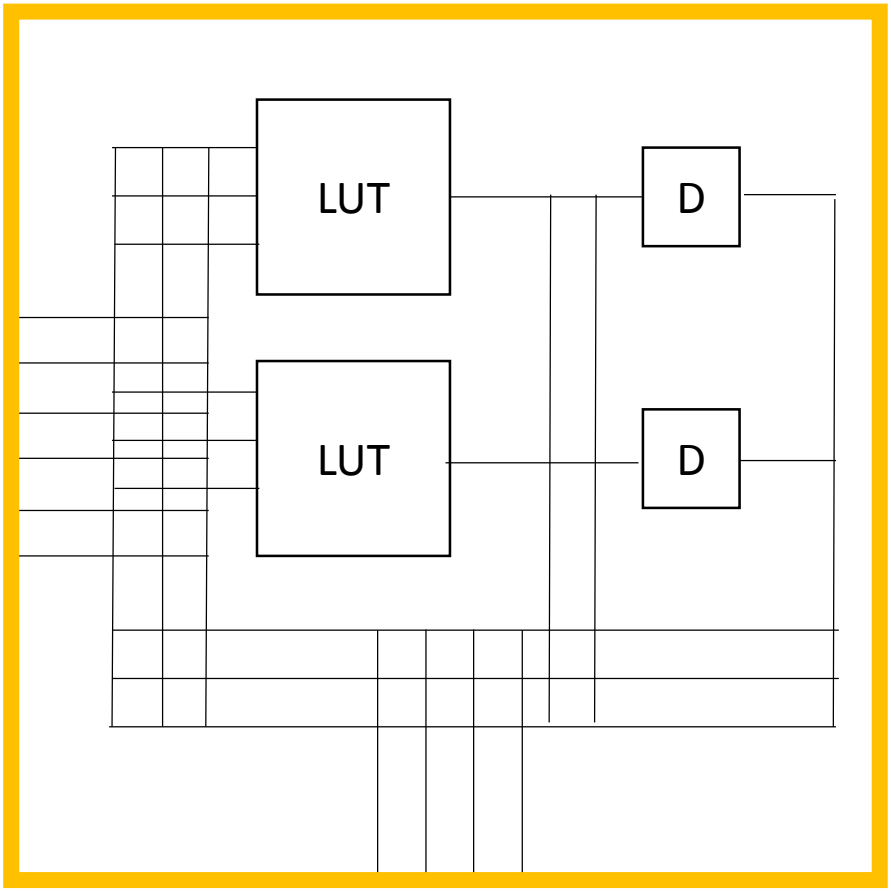
Hierarchické propojení

- Univerzální programovatelné propojení
- Ideálně propojení všeho se vším
- Reálně kompromis se složitostí
- Nástroj provádějící „place“ fázi implementace obvodu (placer) musí znát konkrétní možnosti propojení a musí je co nejvhodněji použít
  - Jde o jednu z výpočetně nejsložitějších fází implementace
- Hodně bitů konfigurace

Hierarchické propojení

Logický blok

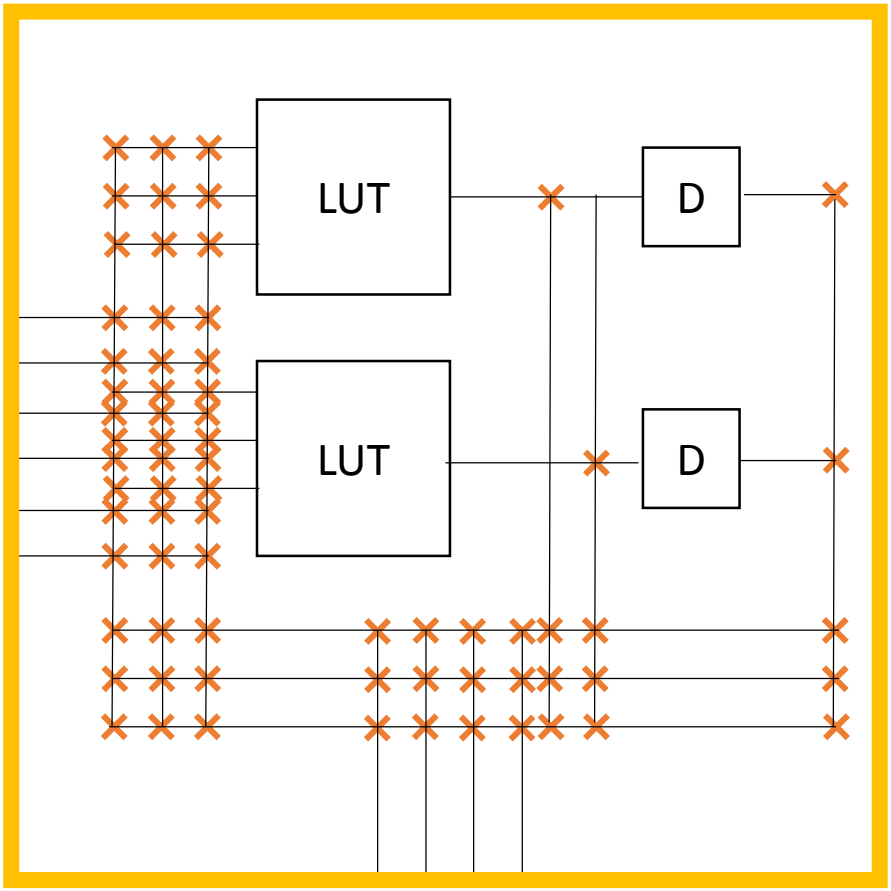
Konfigurace funkce



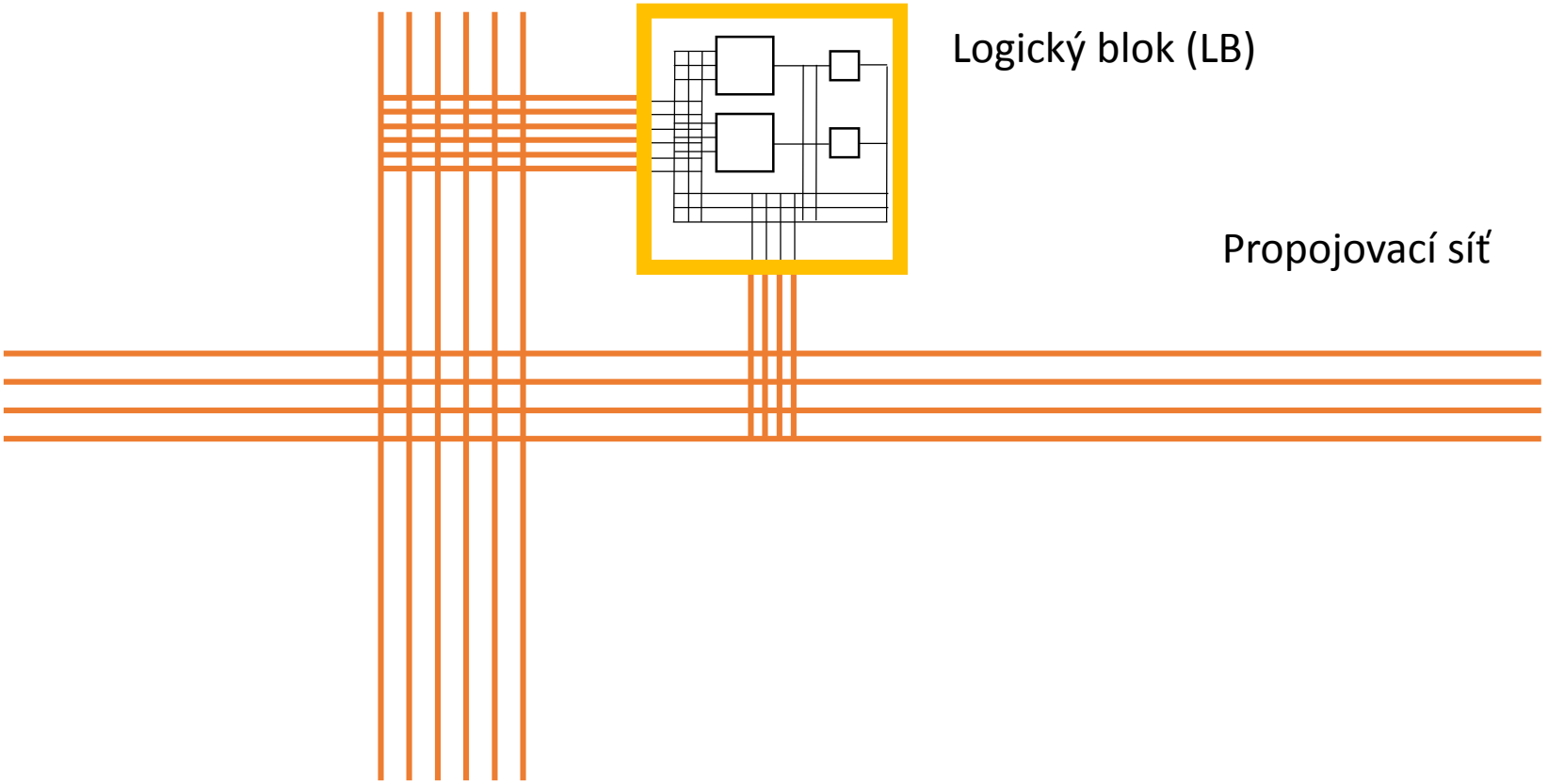
Hierarchické propojení

Logický blok

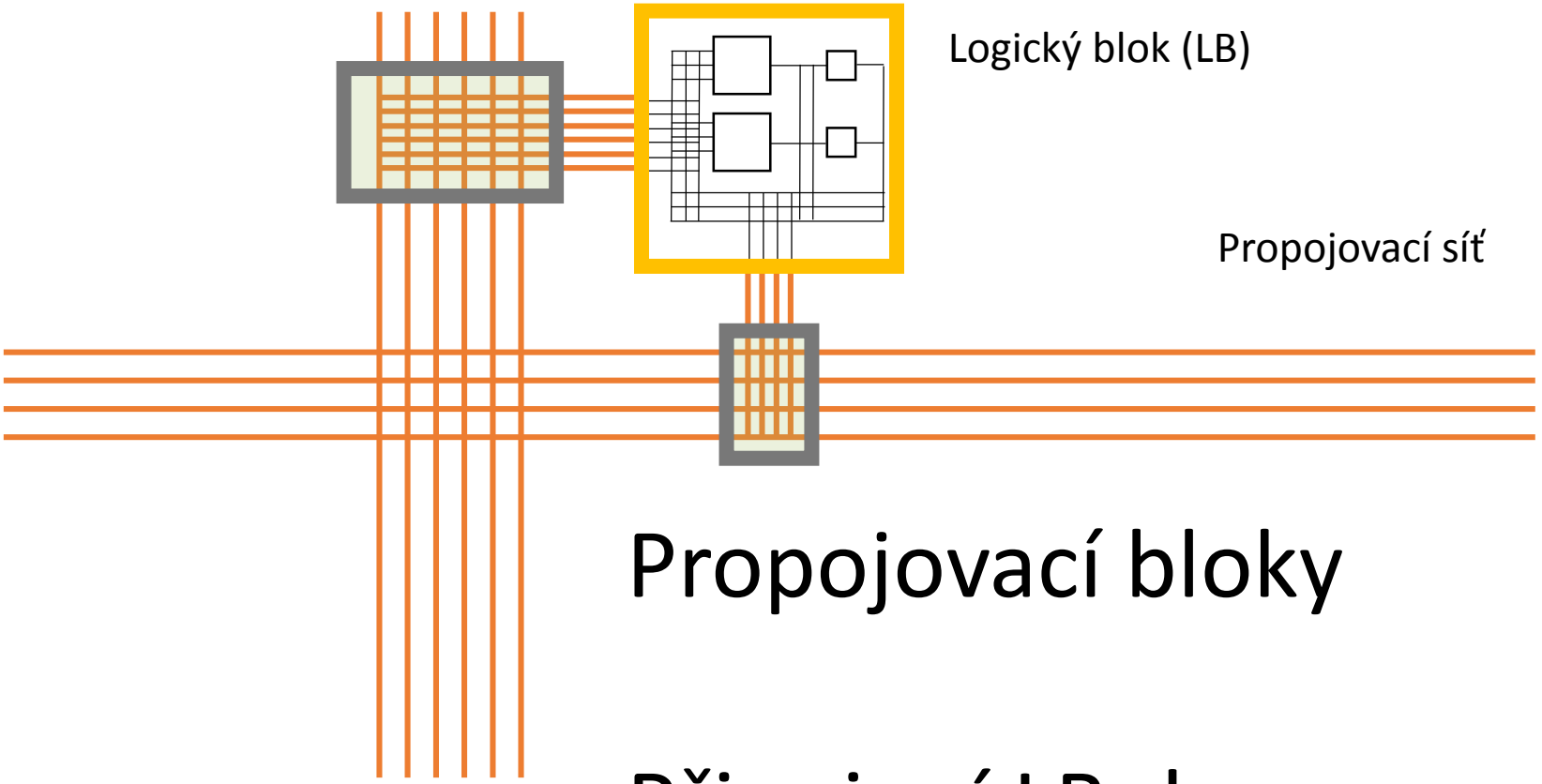
Konfigurace funkce



Hierarchické propojení



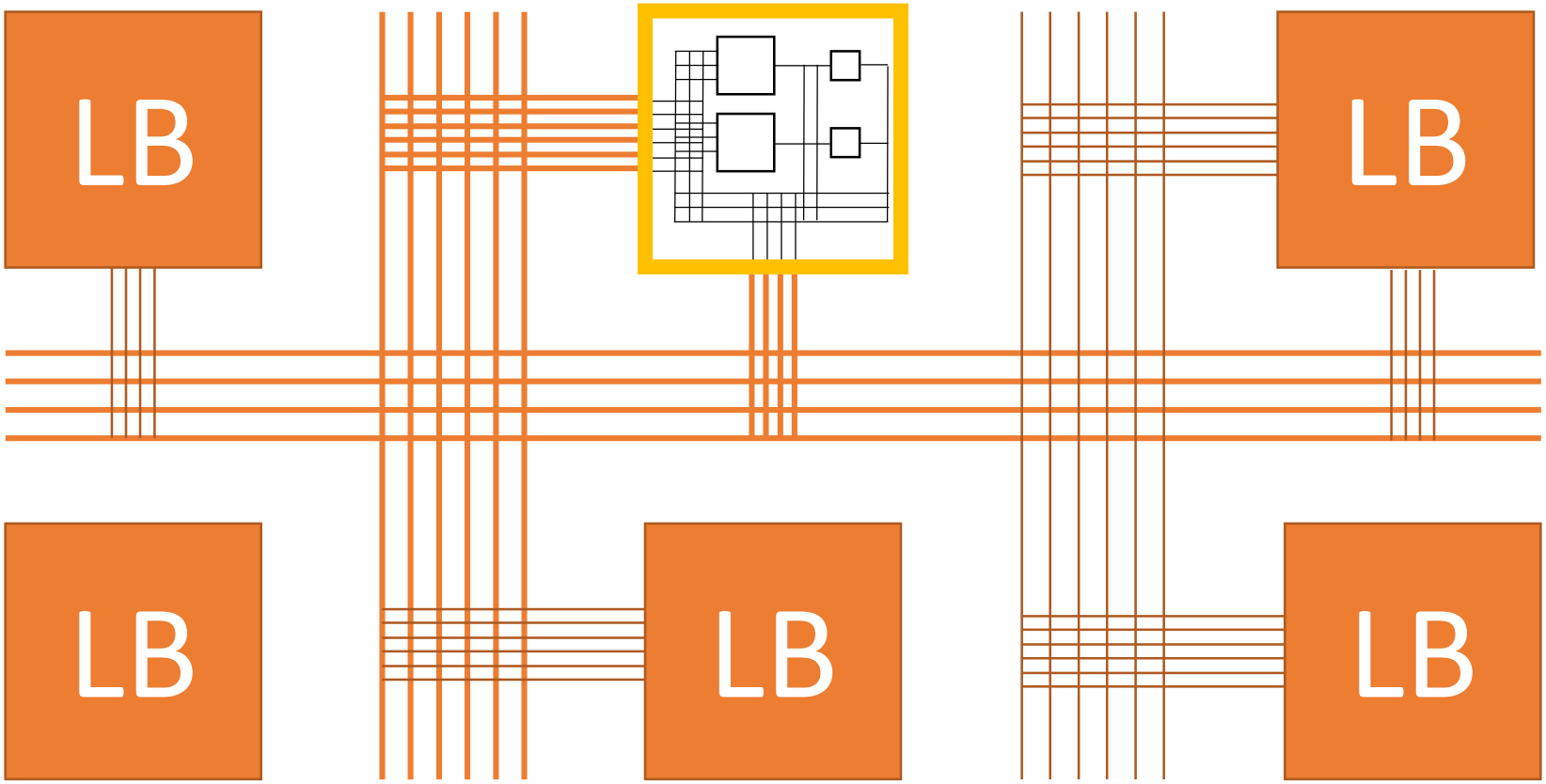
Hierarchické propojení



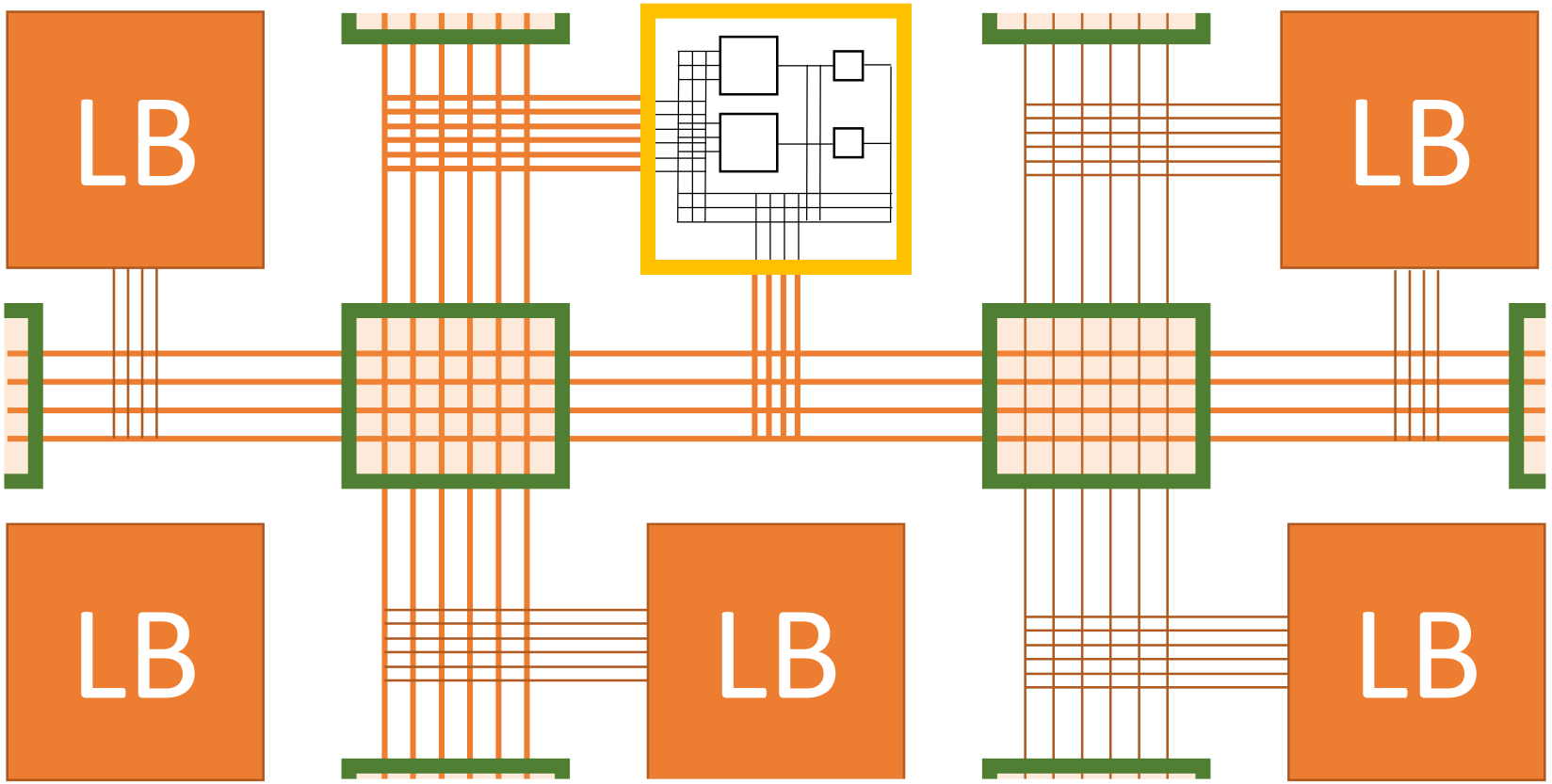
Propojovací bloky

Připojení LB do propojovací sítě

Hierarchické propojení



Hierarchické propojení



Switch blocks – globální propojení

Konfigurační paměť

- Konfigurační paměťové buňky všech vnitřností
- Různá technologie u různých FPGA/výrobců
  - SRAM – stejný CMOS proces jako zbytek čipů – levné, ale po zapnutí se musí nahrát z externího zdroje
  - Flash – dražší proces, menší FPGA, ale pamatuje si i přes vypnutí, je radiačně odolnější než SRAM
  - Antifuse – jednorázové pojistky, speciální FPGA (radiačně nejodolnější)
- Pro „programátora“ je průhledná, ve valné většině případů se s ní v obvodu nemusíme zabývat

Bloky dalších funkcí

- Jednoúčelové funkce „pálené do křemíku“
- Šetří zdroje a jsou rychlejší
- Názvy občas závislé na výrobcích / řadách

- JTAG (Joint Test Action Group, IEEE 1149.1)
  - Základní komunikace s FPGA - programování, testování
- SPI (Serial Peripheral Interface)
  - Rozhraní pro nahrávání konfigurační paměti z externího úložiště po startu FPGA

Bloky dalších funkcí (pokračování)

- PLL / DLL / DCM / MMCM / …
  - Obvody pro práci s hodinami: syntéza frekvence, fázový posun, čištění jitteru, generování více hodin se vzájemným vztahem
- BRAM (Block RAM)
  - Bloky pamětí, velikost v řádu kbitů
- DSP (Digital signal processing)
  - Obvody pro rychlé početní operace
  - MAC (Multiply–accumulate operation), floating point, …
- CPU
  - Např. 2 jádra ARM vypálená na stejném čipu

Ukázka v Xilinx ISE

- Znovu příklad se sčítačkou/čítačem
- Nahlížení do protokolů
- Nahlížení do výsledků jednotlivých kroků
- Prohlížení implementovaného obvodu v čipu

Závěrečné slovo

business.mega.mu

O čem by se dalo (příště) mluvit

- Timing, nebo-li časování
- Pravidla návrhu
  - Hodiny jsou svaté
  - Synchronizace a metastabilita
  - Reset
- Jak psát kód – kdo mu má rozumět (člověk i frontend)
  - Standardy, podpora v toolech
- IP cores, SoC, uBlaze, Picoblaze, HLS
- Opensource HW vs. opensource design flow nástroje
- Simulace, ladění
- Devboardy

Materiály a zdroje

- http://fpga.cz/
- http://vhdl.cz/
- http://opencores.org/
- http://www.ohwr.org/
- http://www.fpga4fun.com/
- http://wavedrom.com/
- Stránky a dokumentace čipů / nástrojů

A co dál?

- Otázky?
- Kdo ještě nespí, může zůstat na rozšířenou ukázku
  - Psaní kódu
  - Procesor v FPGA
- Po skončení lze volně navázat exkurzí do bastlírny
- Feedback: macgyver@siliconhill.cz