



FPGA a vyhledávání v textu

Ondřej Sýkora

Úvod – FPGA

- Opakování z loňského roku
 - Programovatelný hardware
 - FPGA, CPLD
 - Konfigurovatelné obvody – hradla, spoje, registry, ...
 - „Logické jednotky“
 - Konektory, tlačítka, diody, „paměť“, ...
 - Program = konfigurace logických jednotek a jejich zapojení
 - Verilog, VHDL
 - Jazyky pro popis (chování) obvodů
 - Možnost popisu na různých úrovních (algoritmy, hradla, propojení, ...)

Úvod - FPGA

- Opakování z loňského roku
 - Verilog
 - Základní konstrukce
 - Modul – „obvod“
 - Vstupy, výstupy, vnitřní stav, kód pro obsluhu událostí
 - Registr – „proměnná“
 - Číselné hodnoty, pole, asociativní pole (!), ...
 - Spoje – „dráty“
 - Přenos hodnot ze vstupního konce na výstup
 - Propojení obvodů, obvodů s registry, ...
 - Handlery – „procedury, funkce“
 - Spouštění jako reakce na událost (změnu vstupu)

Úvod – FPGA

- Opakování z loňského roku
 - „Workflow“ při práci s FPGA
 1. Zadání – popis požadovaného obvodu
 2. Návrh obvodu ve Verilogu/VHDL
 3. Simulace, testování
 4. Překlad/syntéza pro konkrétní zařízení
 5. Simulace, testování
 6. Upload na zařízení
 7. Skutečný provoz

Úvod – Aplikace

- Aplikace z loňského roku
 - „Knight rider“ a další blikající diody
 - Stavové automaty
 - Automat Aho-Corasickové a vyhledávání v sekvenci stisknutých tlačítek

Vyhledávání v textu

- Možné přístupy
 - Knuth-Morris-Pratt, Shift-Or
 - Vyhledávací (regulární) automaty
 - Deterministické
 - Ručně navržené
 - Algoritmus Aho-Corasickové
 - Generuje automat z množiny slov
 - Nedeterministické
 - Ručně navržené
 - Vygenerované z množiny slov, z regulárních výrazů

Vyhledávání „v Hardware“

- Motivace
 - Aplikace v počítačových sítích
 - Intrusion Detection and Prevention, Firewally, ...
 - Řešení pomocí „univerzálního“ počítače jsou příliš nákladná
 - Přímé zapojení do počítačové sítě
 - Paralelní zpracování
 - Více nezávislých obvodů = hledání dle více kritérií
 - Rychlý výpočet „přímo na železe“

Vyhledávání „v Hardware“

- Způsob použití
 - „Přípravná fáze“
 - Identifikace hledaných výrazů
 - Konstrukce automatu (tvorba kódu)
 - Kompilace pro konkrétní zařízení
 - Upload do zařízení
 - „Běhová fáze“
 - Provoz automatu v konkrétním prostředí
 - Pomalá × rychlá fáze

Co se podařilo

- Deterministické automaty
 - Jednoduchá, přímá implementace
 - Jeden registr pro uložení stavu
 - Přejechody jako vnořené switche

```
case (Key)
```

```
  1: case (State)
```

```
    0:      State = 2;
```

```
    5:      State = 3;
```

```
    6:      State = 2;
```

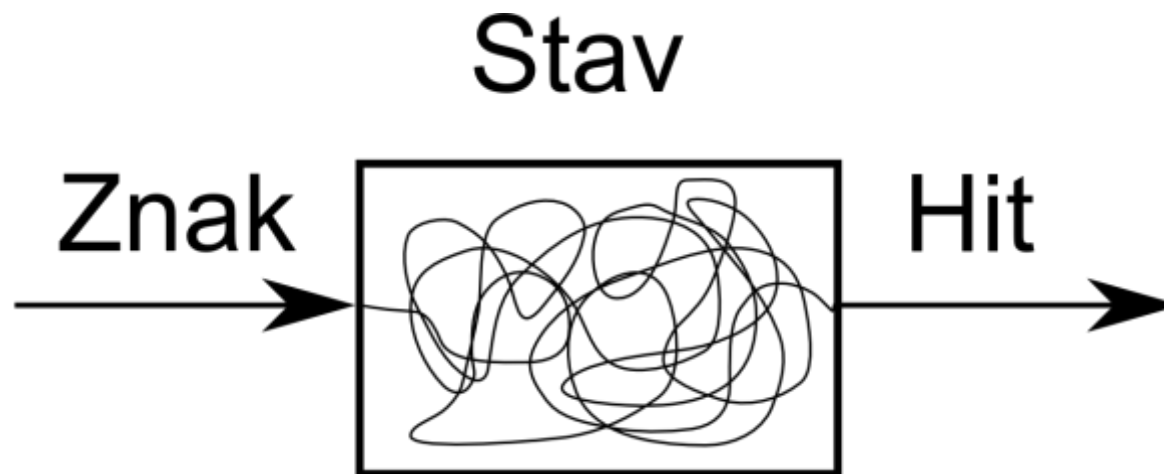
```
  endcase
```

```
endcase
```

- „Snadno“ laditelné – jeden modul, jeden registr, ...

Co se podařilo

- Deterministické automaty
 - Generátor kódu automatu Aho-Corasickové
 - Pevná kostra + přechody podle hledaných slov

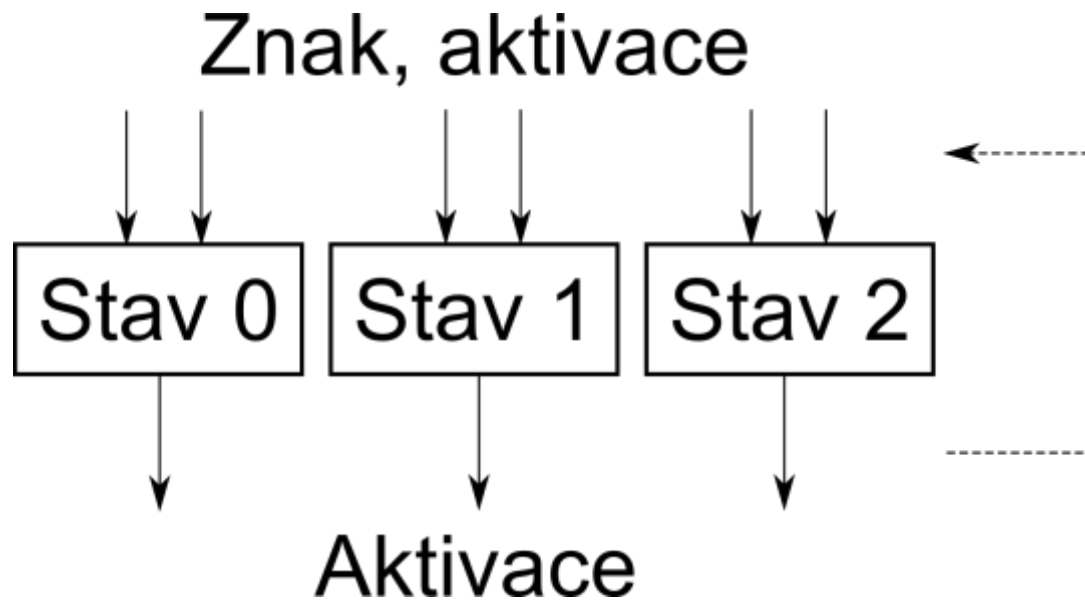


Co se podařilo

- **Nedeterministické automaty**
 - **Obtížnější úloha**
 - Jeden stavový registr nestačí
 - → pole stavových registrů
 - Propojení registrů a kódu
 - → omezený přístup do pole z více procedur
 - Cyklické závislosti stavových registrů
 - → nelze použít model „výpočet jako reakce na vstup“
 - Složitější propojení vstupů, registrů a handlerů
 - → hodnoty stavů je nutné měnit organizovaně

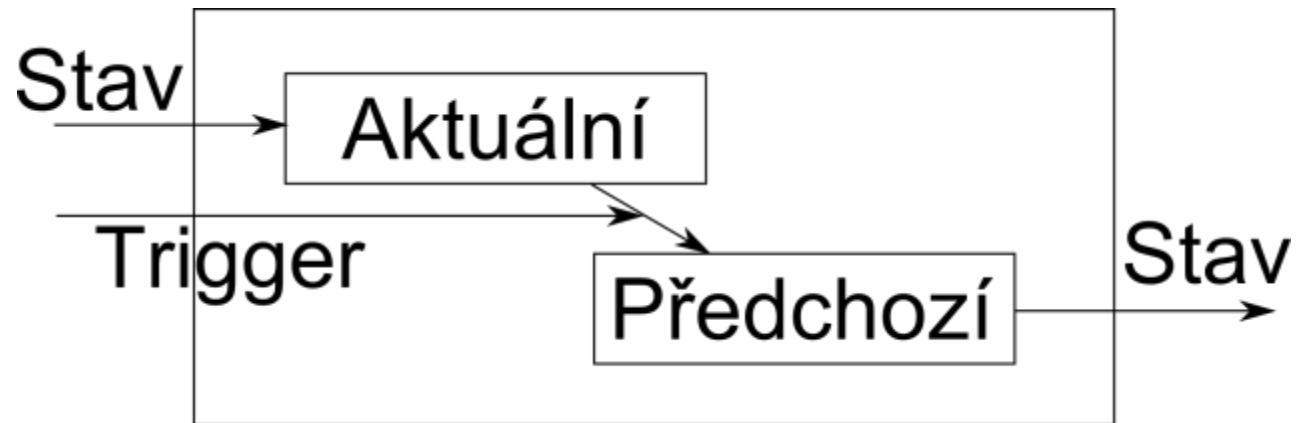
Co se podařilo

- **Nedeterministické automaty**
 - Ručně vytvořený a odladěný automat, zatím bez generátoru



Co se podařilo

- Nedeterministické automaty
 - Modul pro jeden stav, synchronizace



Efektivita

- Srovnání výkonu s implementací na PC
 - Core2 2,4 GHz, Windows Vista x64
 - Deterministický automat na PC

```
for(int i = 0; i < STRING_LENGTH; i++) {  
    state = automat[state][test_string[i]];  
    found |= state == 0;  
}
```

Efektivita

- Srovnání výkonu s implementací na PC
 - DFA, 8 stavů, 4 znaky
 - PC: cca 2,5 ns pro zpracování jednoho znaku
 - Data 400MB řetězec v paměti, čtení je započítáno
 - FPGA: cca 10 ns maximální čas „reakce na vstup“
 - Pouze zpracování změny stavu, nezahrnuje získání dat (čtení z paměti)
 - Složitější DFA, 256 stavů, 64 znaky
 - PC: cca 5,5 ns pro zpracování jednoho znaku
 - FPGA: zatím bez testů
 - Složitost závisí na počtu stavů a velikosti abecedy

Efektivita

- Efektivita NFA
 - Triviální NFA: 4 znaky, 4 stavy
 - Zatím bez implementace na PC
 - Vyžaduje synchronizaci pomocí „externích hodin“, využívá hodiny v FPGA (50 MHz ~ 20 ns)
 - 2 cykly hodin na zpracování jednoho znaku (40 ns)
 - Poroste se složitostí obvodu – závisí na počtu (složitosti) přechodů mezi stavy a velikosti abecedy

Výzvy vývoje pro FPGA

- Verilog + FPGA
 - Verilog – jazyk pro **popis** chování
 - Řada konstrukcí jazyka není podporována na FPGA
 - Dokumentace, specifikace...
 - Výstup kompilátoru, optimalizace
 - Ladění
 - Pouze výstupy simulace, po „optimalizaci“
 - Bez krokování, bez funkce „explain“
 - Hardware
 - Simulace **není** skutečný hardware!

Výzvy vývoje pro FPGA

- Low-level problematika
 - Vyžaduje low-level přístup
 - Př. přístup do paměti
- Knihovny existujících komponent
 - Licence a dostupnost
 - Přehled
 - Dokumentace
 - Potřebné znalosti

Možnosti dalšího rozvoje

- Dokončení testů a měření
 - Generátor kódu NFA pro FPGA
 - Simulace + efektivita NFA na PC
 - Automatizace simulací FPGA
- Prakticky využitelný kód pro FPGA
 - Práce s pamětí na FPGA
 - Přenos dat do/z FPGA

Závěr, dotazy, diskuze

- Zde je prostor pro Vaše dotazy