Rychlé vyhledávání regulárních výrazů s využitím technologie FPGA

Jan Kořenek
Schůzka týmu ANT at FIT

Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 00 Brno, CZ
http://fit.vutbr.cz/~korenek/

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Motivace

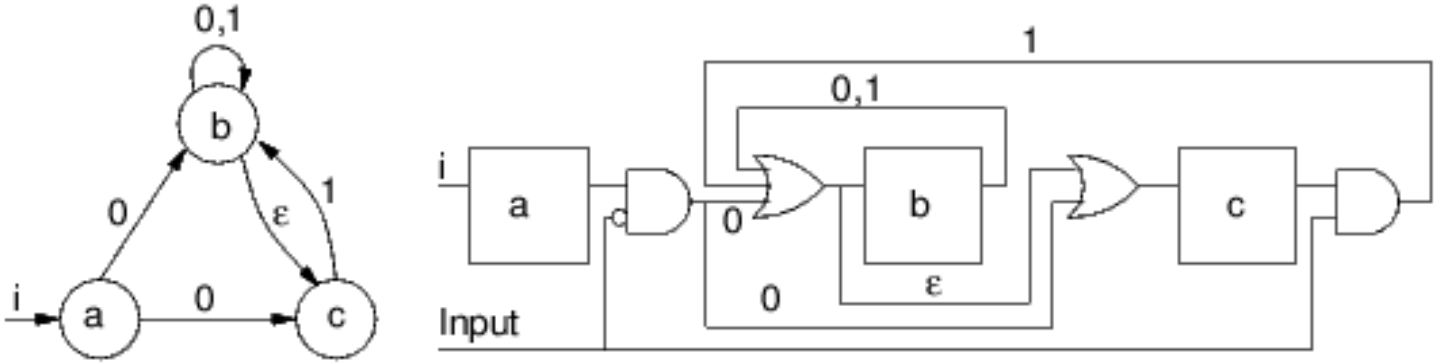
- Hledání regulárních výrazů se používá pro detekci virů, červů a jiného podezřelého chování na síti (IDS, IPS, UTM a další.).
- Časově kritická operace, která u IDS systém Snort spotřebuje přes 90 % výpočetního času procesoru.
- Softwarové implementace dosahují propustnost v řádech stovek megabitů za sekundu.
- Narůstá kapacita síťových linek, ale i počet útoků, virů a jiných bezpečnostních incidentů.
  - ***Hledání tisíců regulárních výrazů***
  - ***na multi-gigabitových rychlostech***
- Je proto snaha urychlit časově kritickou operaci pomocí vhodné hardwarové architektury

Současné hardwarové architektury

- Architektury založené na deterministickém automatu (DFA)
  - Zpracování vstupního symbolu v konstantním čase – garantovaná propustnost
  - Je potřeba provést transformaci na deterministický automat, což může způsobit až exponenciální nárůst tabulky přechodů → ***vysoké požadavky na kapacitu paměti pro uložení tabulky přechodů***
  - Rozdělení regulárních výrazů na podmnožiny
  - Upravené modely automatů – D^2 DFA, H-FA, X-FA, atd.
- Architektury založené na nedeterministickém automatu (NFA)
  - ***Nenarůstá tabulka přechodů***, ale zpracování vstupního symbolu v konstantním čase je možné zajistit současnou aktivací více stavů – ***paralelním zpracováním***
  - Stupeň paralelního zpracování závisí na konkrétním automatu → ***je nutné použít mapování na rekonfigurovatelné struktury jako je FPGA***
  - S velikostí automatu narůstají i požadavky na hardwarové prostředky → ***vysoké požadavky na kapacitu FPGA pro reprezentaci automatu, snaha o redukci***

Mapování NFA do FPGA

- Mapování nedeterministického automatu do FPGA
  - Ke každému stavu je přiřazen jednobitový registr tak, aby bylo možné aktivovat více stavů současně
  - Přechody řešeny pomocí kombinační logiky – komparátor vstupního symbolu, logický AND s výstupním stavem přechodu, logický OR mezi vstupními přechody



- Optimalizace mapování
  - Sdílený dekodér znaků redukuje počet komparátorů (Clark)
  - Sdílení prefixových, infixových a postfixových výrazů (Lin)
- ***S nárůstem počtu regulárních výrazů je potřeba hledat další optimalizace s ohledem na zdroje FPGA***

Efektivita mapování NFA do FPGA

- Kolik procent čipu se uplatní při výpočtu následujícího stavu?
- Analýza kolik stavů může být současně aktivních
  - Převedení množiny regulárních výrazů na NFA a pak na DFA
  - Každý stav S^{DFA} deterministického automatu je reprezentován množinou stavů $\{S^{NFA}_1, S^{NFA}_2, \ldots, S^{NFA}_k\}$ nedeterministického automatu
  - Stav S^{DFA}_{max} definovaný největší množinou stavů odpovídá maximálnímu počtu současně aktivních stavů NFA

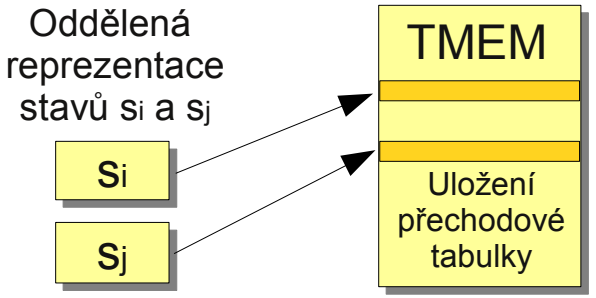
	L7 dekodér	Snort 1	Snort 2	Snort 3	Snort 4	Snort 5
Celkem stavů NFA [-]	774	3888	2774	1060	1038	819
Max. Aktivních stavů [-]	23	122	19	18	25	32
Max. Aktivních stavů [%]	2,97	3,14	0,68	1,7	2,41	3,91

- U všech množin regulárních výrazů méně než 4 % stavů automatu
  - Jen malá část logiky FPGA je v daném okamžiku použita k určení následujících stavů automatu → ***není nutné reprezentovat každý stav automatu jednobitovým registrem***

Redukce zdrojů na čipu

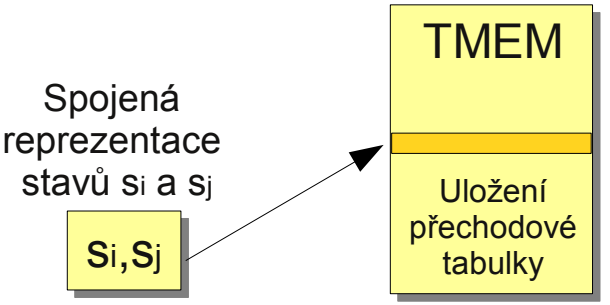
- Pokud je část přechodové tabulky uložena v paměti, je možné redukovat počet využitých zdrojů na čipu (LUT a FF)
  - Pro každý aktivní stav je potřeba přistoupit do paměti s tabulkou přechodů
    - (a) Pro současně aktivní stavy, které sdílejí stejnou paměť je potřeba udělat serializaci přístupů do paměti → dochází ke snižování propustnosti systému
    - (b) Pokud nejsou stavy současně aktivní, můžu je umístit do stejné paměti aniž by mohlo dojít ke snížení propustnosti systému

Stavy s_i a s_j mohu být současně aktivní



Serializace přístupů do paměti nebo dvě oddělené paměti

Stavy s_i a s_j nemohu být současně aktivní



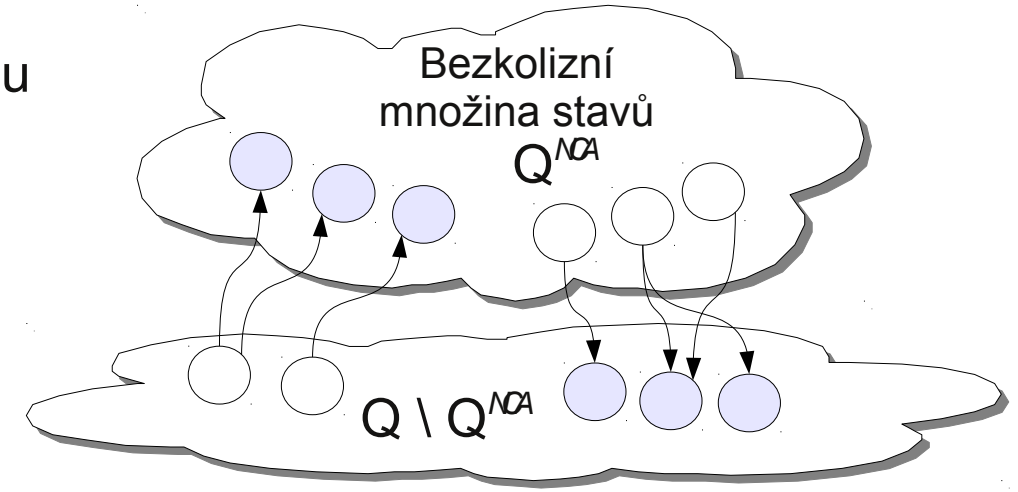
Spojená reprezentace stavů, jedna paměť

- Je potřeba identifikovat v automatu množiny stavů, které nemohou být současně aktivní – bezkolizní množiny stavů

Výpočet bezkolizní množiny stavů

Algoritmus pro nalezení bezkolizní množiny stavů

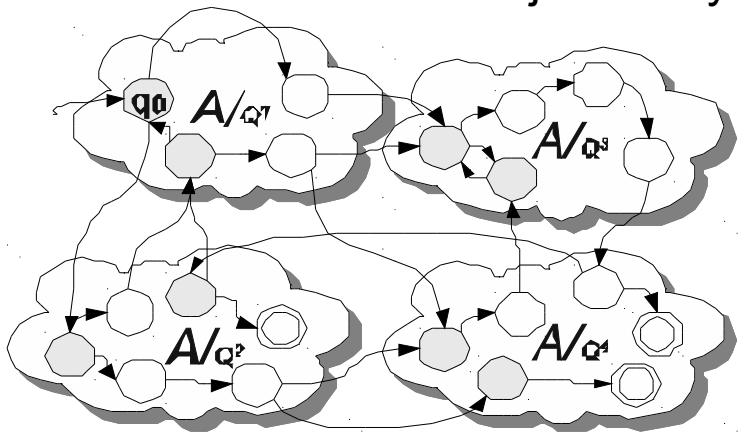
1) Z množiny regulárních výrazů vytvoř nejprve NFA a pak DFA
2) Vypočti pro každý stav $q_i \in Q$ množinu kolizních stavů $S^{CA}_{q_i}$
3) Nechť $Q^{NCA} = Q$
4) Redukuj množinu stavů Q^{NCA}
   - *(a) Vyber stav $q \in Q^{NCA}$ s nejvíce kolizemi odstraň jej z množiny Q^{NCA}*
   - *(b) Pokud množina Q^{NCA} neobsahuje jenom stavy bez kolize, jdi zpět do bodu (a)*
5) Q^{NCA} je množina stavů bez kolize



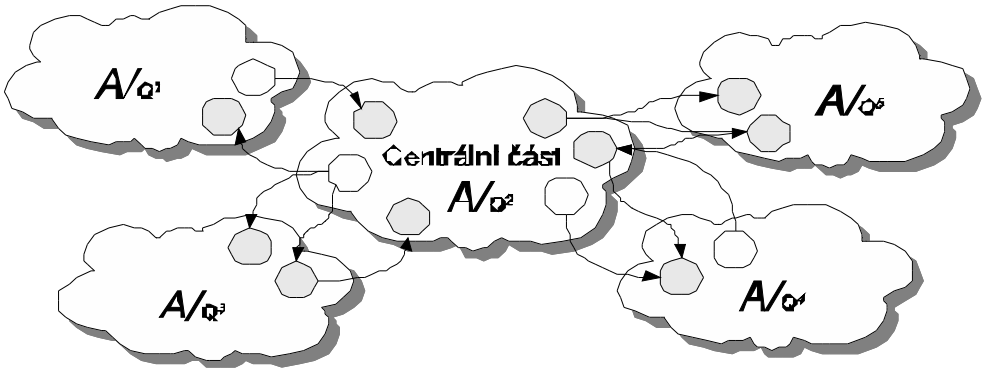
V bezkolizní množině stavů nemohou být žádné dva stavy současně aktivní

Systém paralelních částí automatu

- Na základě rozdělení množiny stavů na podmnožiny je možné definovat Systém paralelních částí automatu
  - Bezkolizní množiny stavů Q^{NCA} určují deterministické části automatu A/Q^{NCA}
  - Množina Q^{N} , ve které může být více současně aktivních stavů, definuje nedeterministickou části automatu A/Q^{N}
- Centralizovaný systém paralelních částí automatu
  - Redukce počtu obousměrných propojů z ***k•(k-1)/2*** na pouhých ***(k-1)***
  - Komunikace mezi jednotlivými části automatu musí procházet přes centrální část



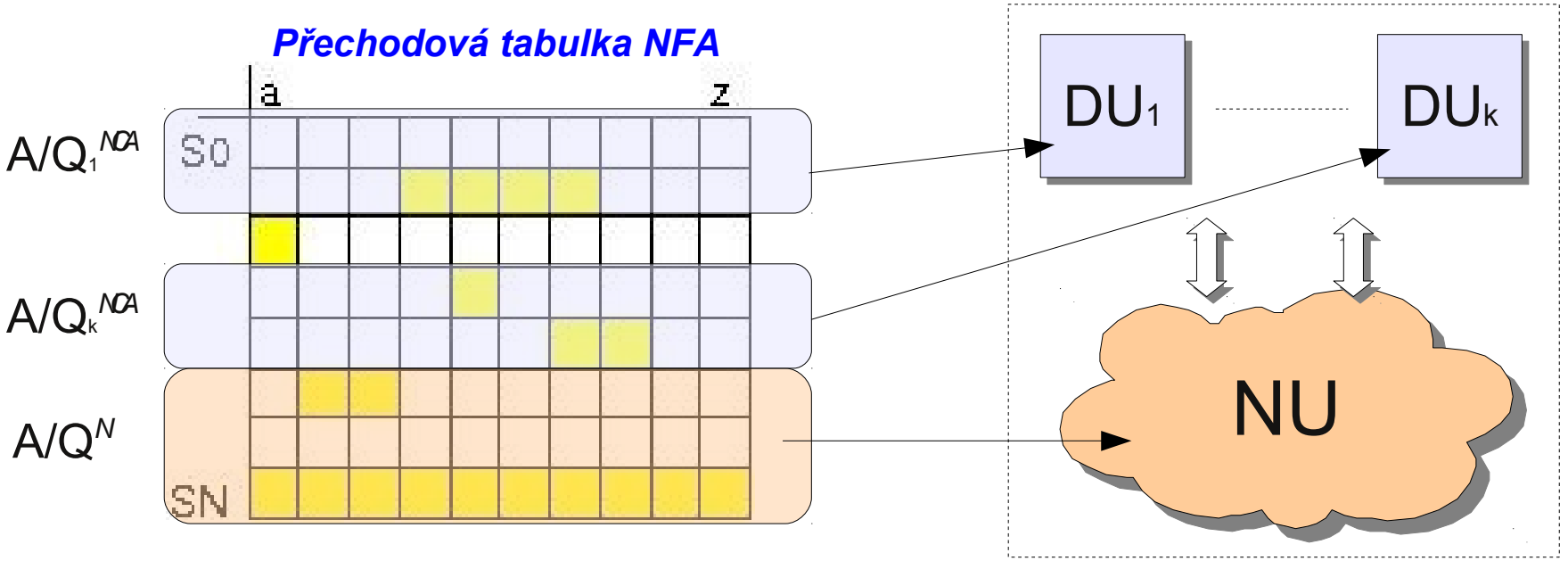
Systém paralelních částí automatu



Centralizovaný systém paralelních částí automatu

Architektura NFA k Split

- **Mapování systému paralelních částí automatu do FPGA**



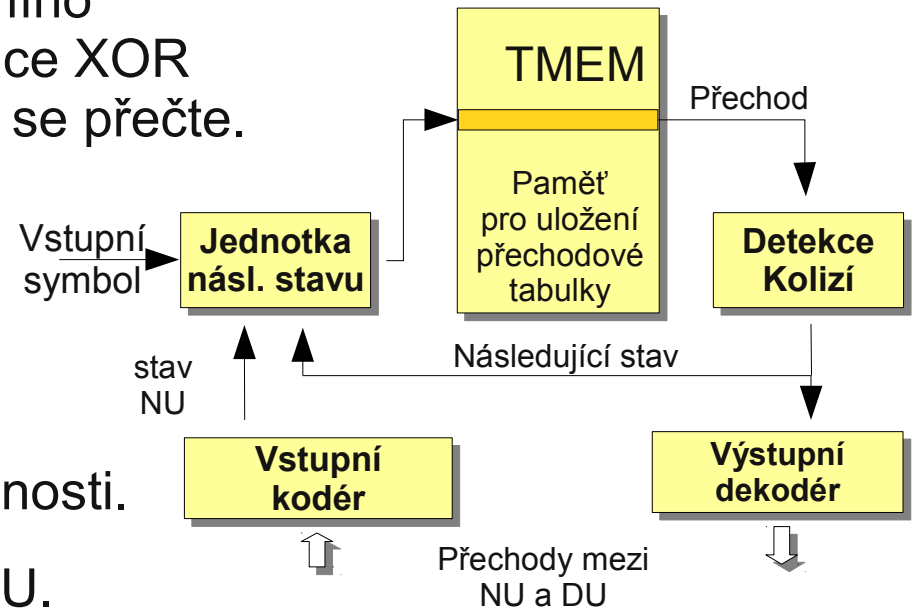
- Deterministické části automatu jsou reprezentovány DU
  - Přechodová tabulka uložena v paměti
  - Redukce počtu registrů pomocí binárního kódování stavů
- Nedeterministická část automatu je mapována do logiky FPGA

Architektura DU

- DU slouží k mapování deterministické části automatu do FPGA.
- Odpovídající část tabulky přechodů je uložena v paměti TMEM.
- Z důvodu úspory paměti jsou řádky tabulky v paměti vzájemně překryty.

Popis činnosti DU

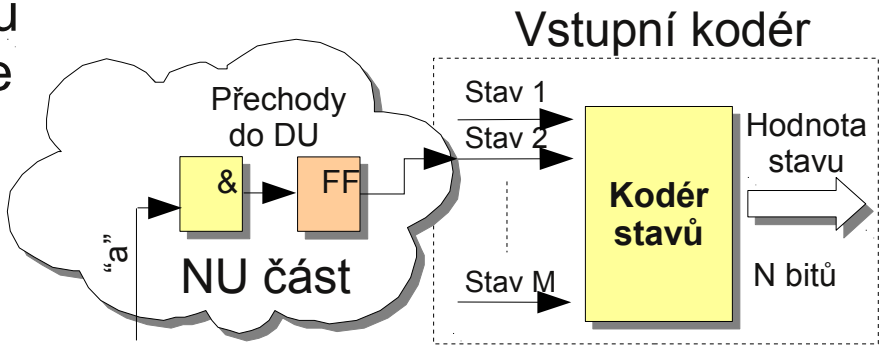
1) Na základě aktuálního stavu a vstupního symbolu se vypočte s využitím operace XOR adresa přechodu v TMEM a přechod se přečte.
2) Provede se detekce kolize s překrytými řádky tabulky.
3) Pokud nedošlo ke kolizi, stává se následující stav stavem aktuálním. Při kolizi přechází DU do stavu nečinnosti.
4) Je možné kdykoliv aktivovat stav z NU.



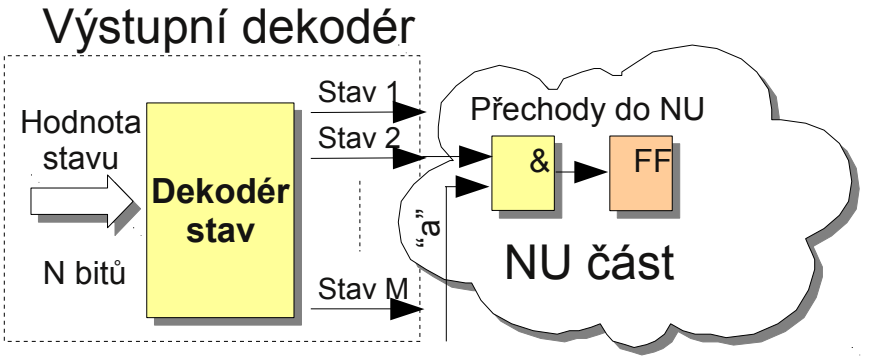
Přechody mezi DU a NU

- Díky přechodům mezi deterministickou a nedeterministickou částí automatu je nutné řešit komunikaci mezi NU a DU
- Díky rozdílnému kódování stavů je nutné použít kodér a dekodér stavů
- Dekodér – s počtem stavů logaritmicky narůstá počet vstupů → není limitujícím prvkem pro dosažení max. frekvence.
- Kodér – s počtem stavů lineárně narůstá počet vstupů → výrazně tak narůstá i počet look-up tabulek v kaskádě → snižuje se max. frekvence

Přechody z NU do DU



Přechody z DU do NU



- Zrychlení kodéru – (1) rozdělení kodéru na dva zřetězené stupně a (2) rozdělení DU na více části.

Dosažené výsledky

- Vyhodnocení algoritmu pro hledání bezkolizních množiny stavů

	NFA stavů	$Q1^{nca}$ [%]	$Q2^{nca}$ [%]	$Q3^{nca}$ [%]	$Q4^{nca}$ [%]	$Q5^{nca}$ [%]	$Q6^{nca}$ [%]	$Q7^{nca}$ [%]	$Q8^{nca}$ [%]	QN [%]
L7 dek.	806	78.40	7.70	4.00	1.70	1.90	0.70	0.50	0.60	4.50
Snort (1)	3888	83.60	6.20	2.60	1.30	0.80	0.70	0.70	0.40	3.70
Snort (2)	819	63.70	8.50	5.00	2.80	2.20	2.00	1.60	1.50	12.70
Snort (3)	527	59.80	7.00	4.90	3.00	2.10	2.10	2.10	2.10	16.90
Snort (4)	1344	35.90	8.80	5.00	2.30	1.60	0.80	0.70	0.50	44.40
Snort (5)	1060	70.70	10.80	5.40	3.50	3.10	1.70	0.90	0.80	3.10
Snort (6)	1038	56.90	18.80	7.50	4.70	3.20	2.40	2.30	0.50	3.70
Snort (7)	2774	53.40	28.40	5.40	4.50	2.30	2.10	1.10	0.90	1.90
Snort (8)	1398	61.00	7.20	3.10	2.00	1.60	1.20	1.10	1.00	21.80
Snort (9)	4513	51.20	1.70	1.30	1.20	1.10	1.10	1.00	1.00	40.40

- První bezkolizní množina obsahuje v průměru 61,46 % stavů NFA a v případě regulárních výrazů Snort(1) dokonce 83,6 % stavů

Využití logiky na čipu

- Syntéza architektury NFA Split do FPGA Virtex-5 LX 155T

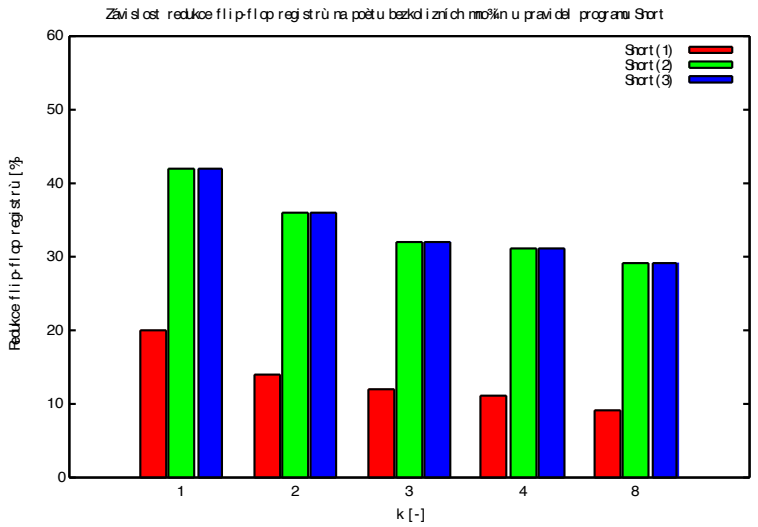
	Clark et al		NFA Split			Reduction	
	LUT [-]	FF [-]	LUT [-]	FF [-]	BRAM	LUT[%]	FF [%]
L7 dekodér	1538	836	1231	237	3	80.04	28.35
Snort (1)	4680	4043	2466	821	21	52.69	20.31
Snort (2)	2965	876	1883	374	15	63.51	42.69
Snort (3)	1637	555	1370	261	6	83.69	47.03
Snort (4)	2436	1392	2233	924	6	91.67	66.38
Snort (5)	2807	1099	1969	368	6	70.15	33.48
Snort (6)	2680	1097	2259	543	6	84.29	49.50
Snort (7)	10314	2812	3393	1439	6	32.90	51.17
Snort (8)	18565	13042	14332	5484	30	77.20	42.05
Snort (9)	65265	40910	29670	20147	39	45.46	49.25
Snort all	**111349**	**65826**	**59575**	**30361**	**135**	**53.50**	**46.12**

V průměru snížila architektura NFA Split u všech množin regulárních výrazů počet look-up tabulek o 66.8% a počet flip-flop registrů o 43.3%

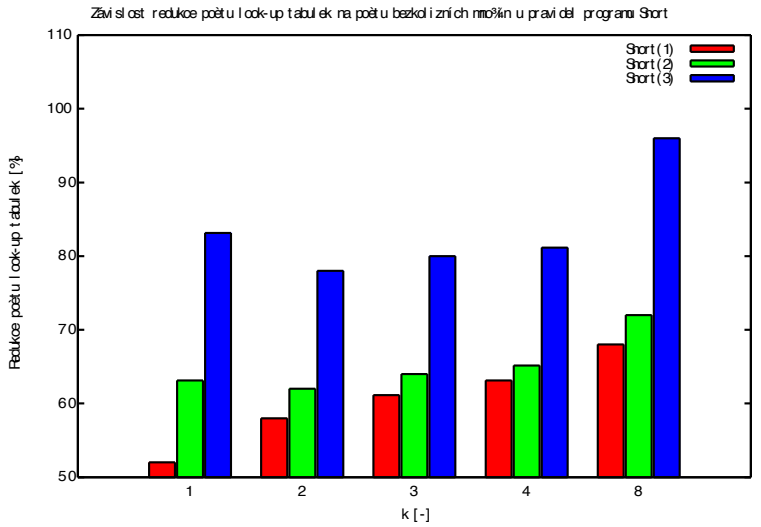
Redukce logiky na čipu

- Redukce počtu flip-flop registrů look-u tabulek pro architekturu NFA k Split

Redukce flip-flop registrů



Redukce look-up tabulek



- Pro k > 3 se projevuje režie komunikace mezi DU a NU a již nedochází k další redukci zdrojů na čipu

Shrnutí

- Na základě analýzy množin regulárních výrazů byl navržen algoritmus pro hledání bezkolizních množin stavů automatu
  - Cílem bylo mapovat část přechodové tabulky do paměti místo do FPGA
  - Algoritmus byl schopen najít množiny obsahující v průměru 61,4 % stavů automatu, v jednom případě dokonce 83,6 % stavů automatu
- Navržen formální model: Systém paralelních částí automatu
- Pro vytvořený formální model byla navržena nová architektura NFA Split a její zobecnění NFA k Split
  - NFA Split redukce v průměru o 66,8 % LUT a o 43,3 % FF.
  - S rostoucím k již nedochází k výraznému zlepšení
  - Zvýšení frekvence s využitím paralelních deterministických jednotek

Konec prezentace

Děkuji za pozornost