Hardwarově akcelerovaná kryptografie s využitím FPGA

Autor: Petr Jedlička
Rok: 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Obsah přednášky

Základní pojmy z oblasti digitálního návrhu

- Elementární bloky v FPGA,
- porovnání s implementací na CPU.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Obsah přednášky

Základní pojmy z oblasti digitálního návrhu

- Elementární bloky v FPGA,
- porovnání s implementací na CPU.

Návrh digitálních funkčních bloků

- Ukázky způsobů paralelizace na konkrétních příkladech,
- návrh komponent pro postkvantovou kryptografii.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Obsah přednášky

Základní pojmy z oblasti digitálního návrhu

- Elementární bloky v FPGA,
- porovnání s implementací na CPU.

Návrh digitálních funkčních bloků

- Ukázky způsobů paralelizace na konkrétních příkladech,
- návrh komponent pro postkvantovou kryptografii.

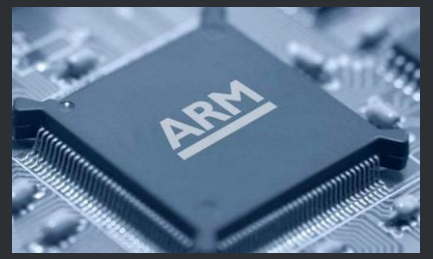
Verifikace na FPGA

- Simulace,
- ILA bloky.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FPGA vs. CPU

- Jazyk pro hardwarový popis (VHDL, Verilog, …),

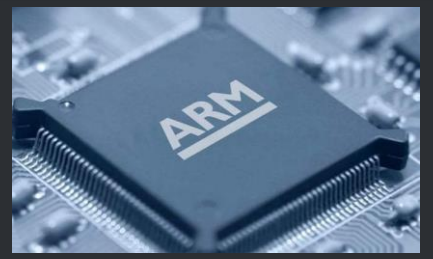


- Programovací jazyk (C, C++, …),

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FPGA vs. CPU

- Jazyk pro hardwarový popis (VHDL, Verilog, …),
- paralelní vykonávání,

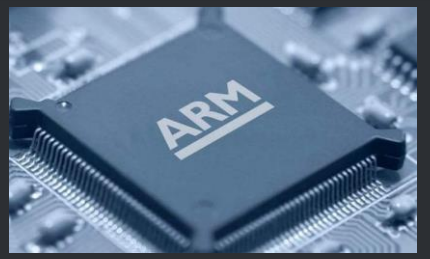


- Programovací jazyk (C, C++, …),
- sekvenční vykonávání,

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FPGA vs. CPU

- Jazyk pro hardwarový popis (VHDL, Verilog, …),
- paralelní vykonávání,
- elementarní digitální prvky,

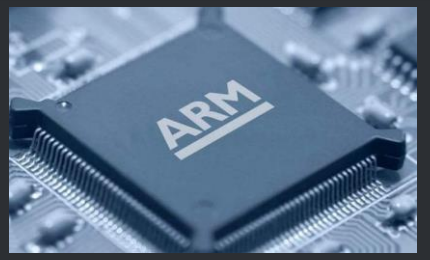


- Programovací jazyk (C, C++, …),
- sekvenční vykonávání,
- sada instrukcí,

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FPGA vs. CPU

- Jazyk pro hardwarový popis (VHDL, Verilog, …),
- paralelní vykonávání,
- elementarní digitální prvky,
- funkcionalita dána propojením digitálních prvků,

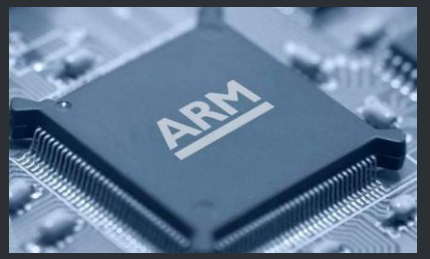


- Programovací jazyk (C, C++, …),
- sekvenční vykonávání,
- sada instrukcí,
- funkcionalita dána strojovým kódem,

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FPGA vs. CPU

- Jazyk pro hardwarový popis (VHDL, Verilog, …),
- paralelní vykonávání,
- elementarní digitální prvky,
- funkcionalita dána propojením digitálních prvků,
- vhodné pro matematické/logické operace s možností paralelizace (DSP, operace s vektory, …) nad velkým množstvím dat,

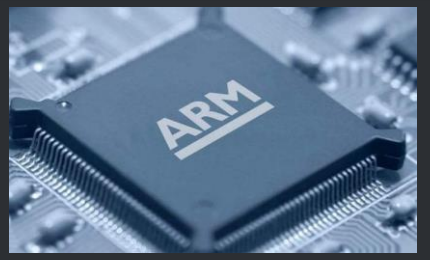


- Programovací jazyk (C, C++, …),
- sekvenční vykonávání,
- sada instrukcí,
- funkcionalita dána strojovým kódem,
- vhodné pro různé rozhodovací algoritmy (parsování dat, …),

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FPGA vs. CPU

- Jazyk pro hardwarový popis (VHDL, Verilog, …),
- paralelní vykonávání,
- elementarní digitální prvky,
- funkcionalita dána propojením digitálních prvků,
- vhodné pro matematické/logické operace s možností paralelizace (DSP, operace s vektory, …) nad velkým množstvím dat,
- efektivnější implementace.

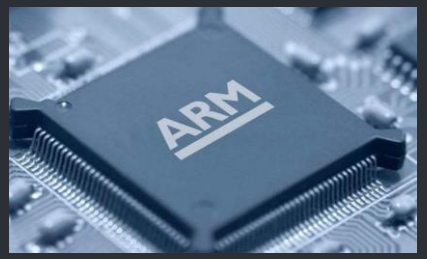


- Programovací jazyk (C, C++, …),
- sekvenční vykonávání,
- sada instrukcí,
- funkcionalita dána strojovým kódem,
- vhodné pro různé rozhodovací algoritmy (parsování dat, …),
- rychlejší vývoj.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FPGA vs. CPU

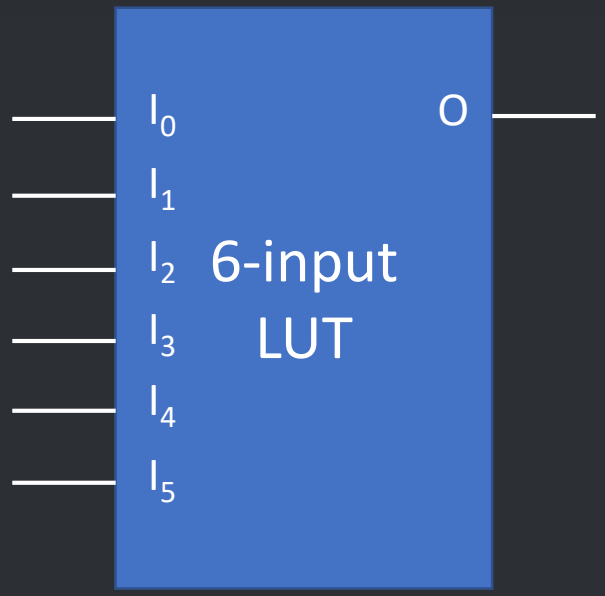
- Jazyk pro hardwarový popis (VHDL, Verilog, …),
- paralelní vykonávání,
- elementarní digitální prvky,
- funkcionalita dána propojením digitálních prvků,
- vhodné pro matematické/logické operace s možností paralelizace (DSP, operace s vektory, …) nad velkým množstvím dat,
- efektivnější implementace,
- upovídanější kód/nižší míra abstrakce (12500 řádků).



- Programovací jazyk (C, C++, …),
- sekvenční vykonávání,
- sada instrukcí,
- funkcionalita dána strojovým kódem,
- vhodné pro různé rozhodovací algoritmy (parsování dat, …),
- rychlejší vývoj,
- vyšší míra abstrakce (2500 řádků).

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

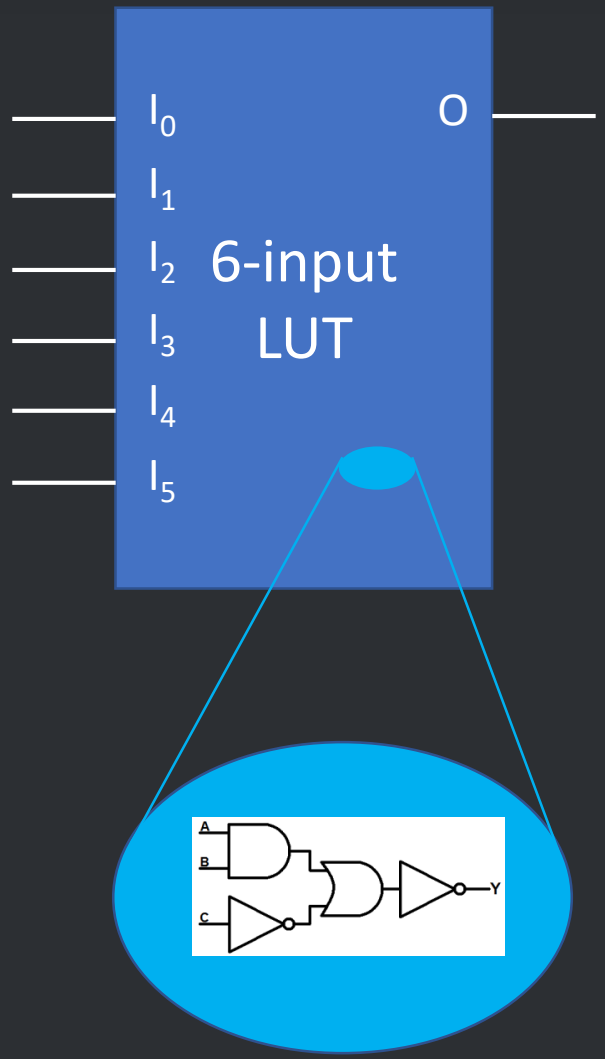
Základní stavební bloky v FPGA



- Lookup tabulky (LUT),

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

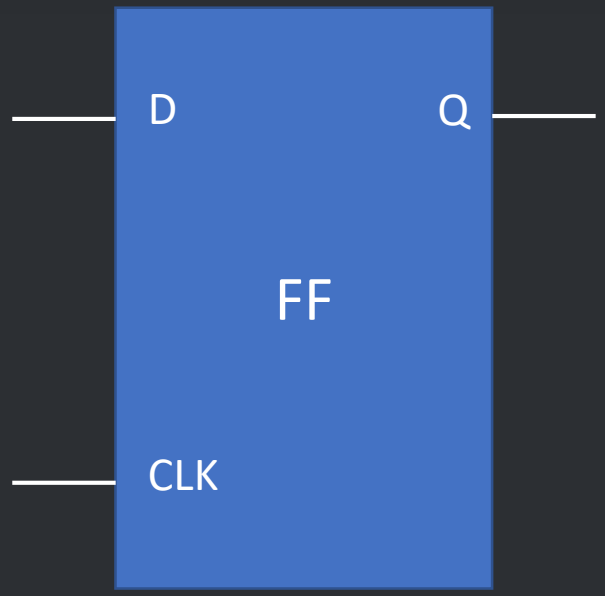
Základní stavební bloky v FPGA



- Lookup tabulky (LUT),

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

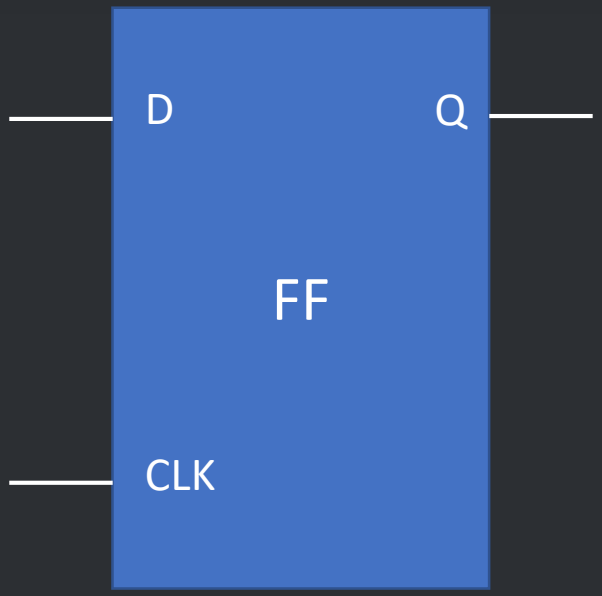
Základní stavební bloky v FPGA



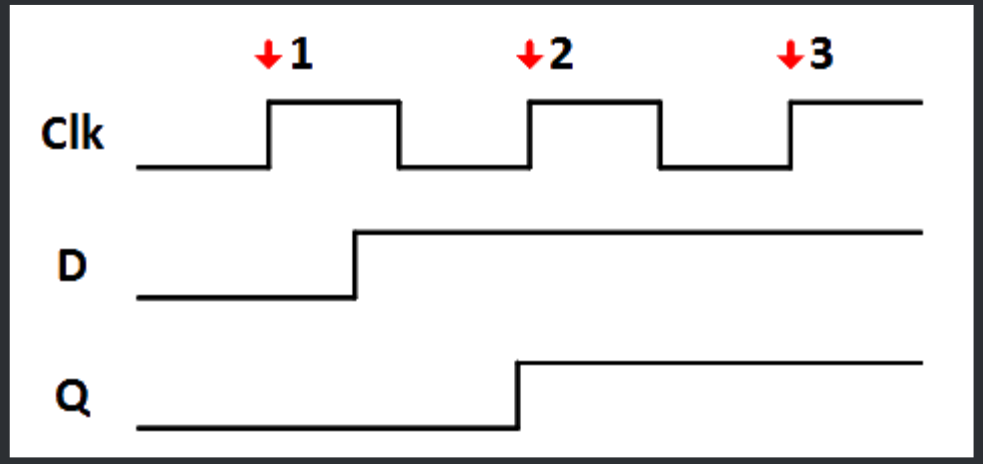
- Lookup tabulky (LUT),
- Flip-flopy (FF),

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Základní stavební bloky v FPGA

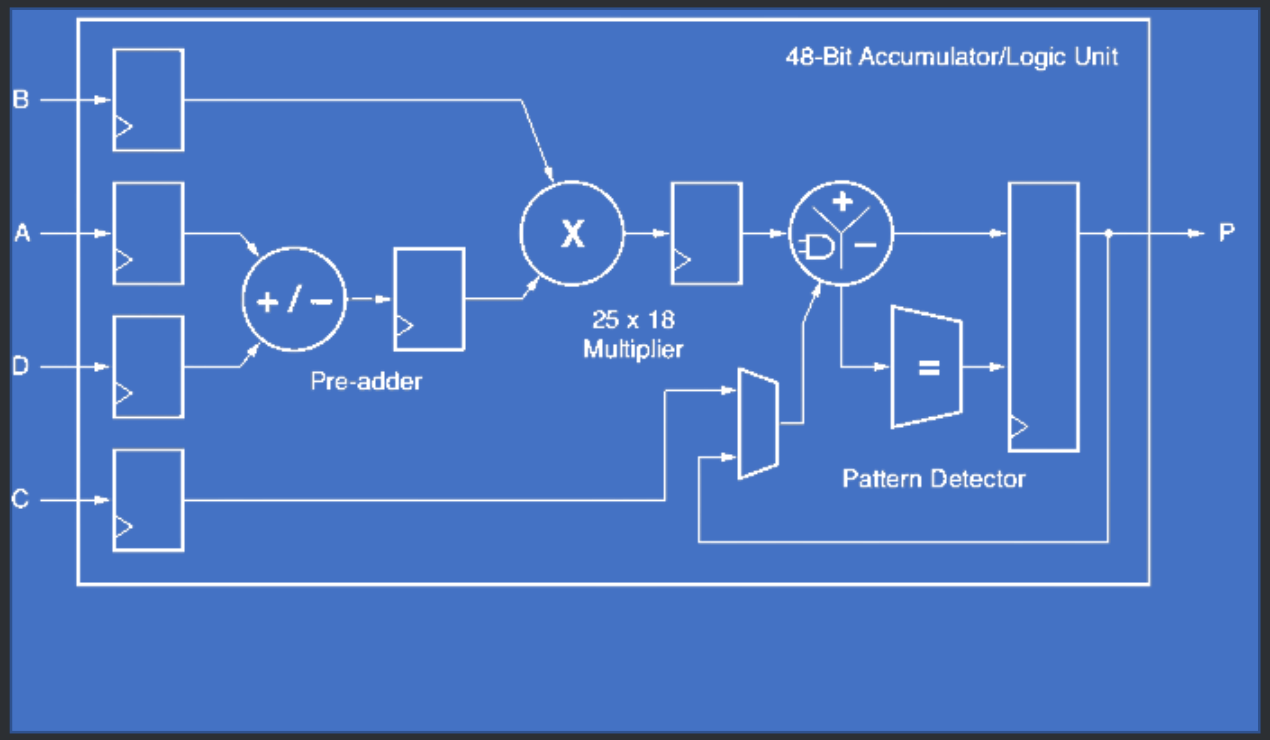


- Lookup tabulky (LUT),
- Flip-flopy (FF),



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

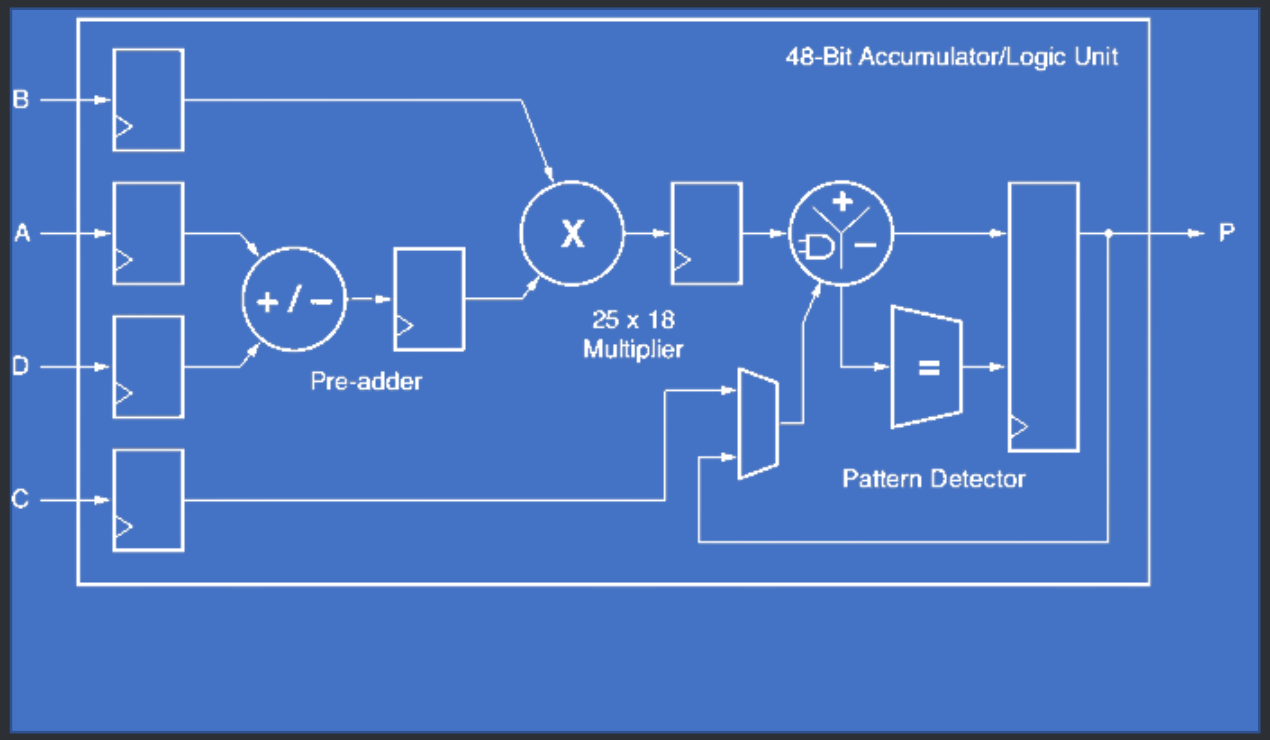
Základní stavební bloky v FPGA



- Lookup tabulky (LUT),
- Flip-flopy (FF),
- DSP48,

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

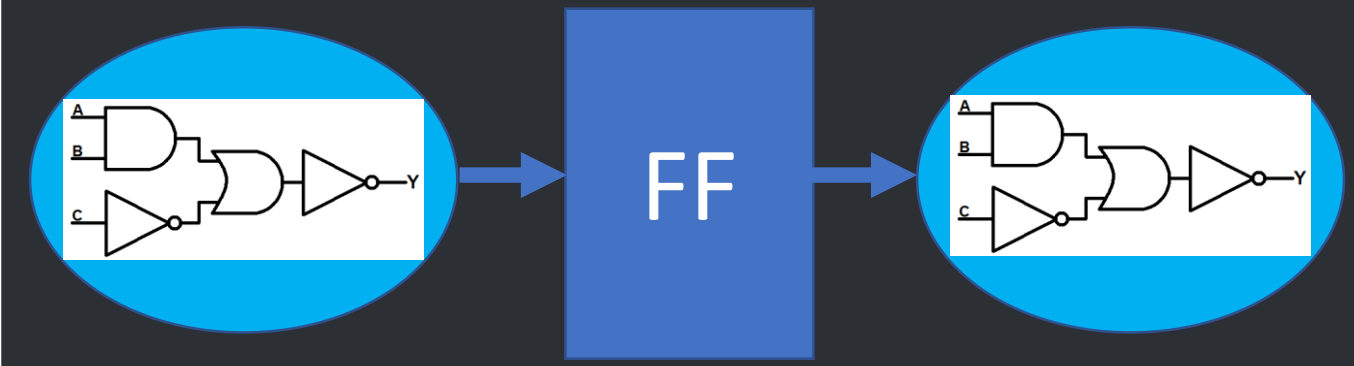
Základní stavební bloky v FPGA



- Lookup tabulky (LUT),
- Flip-flopy (FF),
- DSP48,
- paměťové prvky (BRAM, UltraRAM),
- generování hodinového signálu (PLL, Clock Managery),
- periferie pro komunikační rozhraní (PCIe, QSFP, …)

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

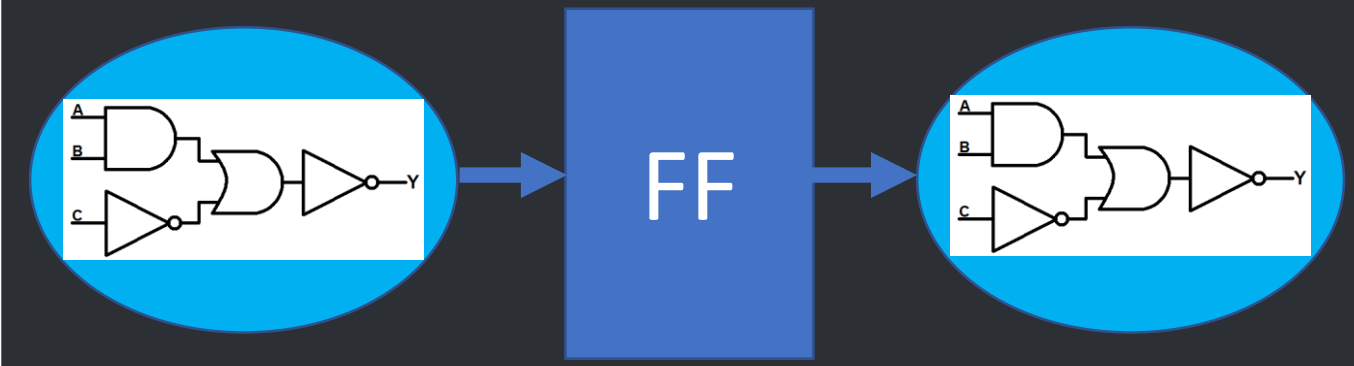
Sledované parametry



- Taktovací frekvence,
- latence,
- propustnost,
- využití hardwarových zdrojů.

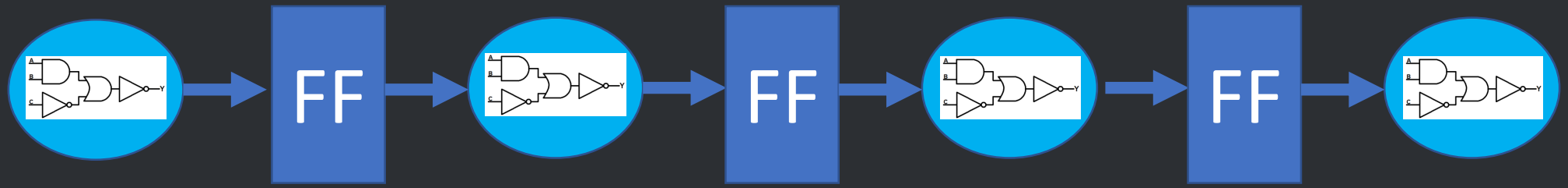
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Sledované parametry



- Taktovací frekvence,
- latence,
- propustnost,
- využití hardwarových zdrojů.

více hw zdrojů,
vyšší latence,
vyšší frekvence,
konstantní propustnost



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Perspektivní kryptografické algoritmy

Finalisté 3. kola NIST standardizace pro postkvantovou kryptografii:

Název schématu	Typ schématu
Šifrování a distribuce klíčů	
Kyber	lattice-based
McEliece	code-based
NTRU	lattice-based
Saber	lattice-based
Digitální podpis	
Dilithium	lattice-based
Falcon	lattice-based
Rainbow	multivariate

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Perspektivní kryptografické algoritmy

Finalisté 3. kola NIST standardizace pro postkvantovou kryptografii:

Název schématu	Typ schématu
Šifrování a distribuce klíčů	
Kyber	lattice-based
McEliece	code-based
NTRU	lattice-based
Saber	lattice-based
Digitální podpis	
Dilithium	lattice-based
Falcon	lattice-based
Rainbow	multivariate

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Lattice-based algoritmy

$$NTT(a) \times NTT(b) = NTT(c)$$

$$p_0x^0 + p_1x^1 + \ldots + p_{255}x^{255}$$

Typické výpočetně náročné operace

- NTT (Number Theoretic Transform),

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Lattice-based algoritmy

$$As + e = t$$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{bmatrix}$$

$$p_0x^0 + p_1x^1 + \ldots + p_{255}x^{255}$$

Typické výpočetně náročné operace

- NTT (Number Theoretic Transform),
- násobení matic a vektorů obsahujících polynomy,

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Lattice-based algoritmy

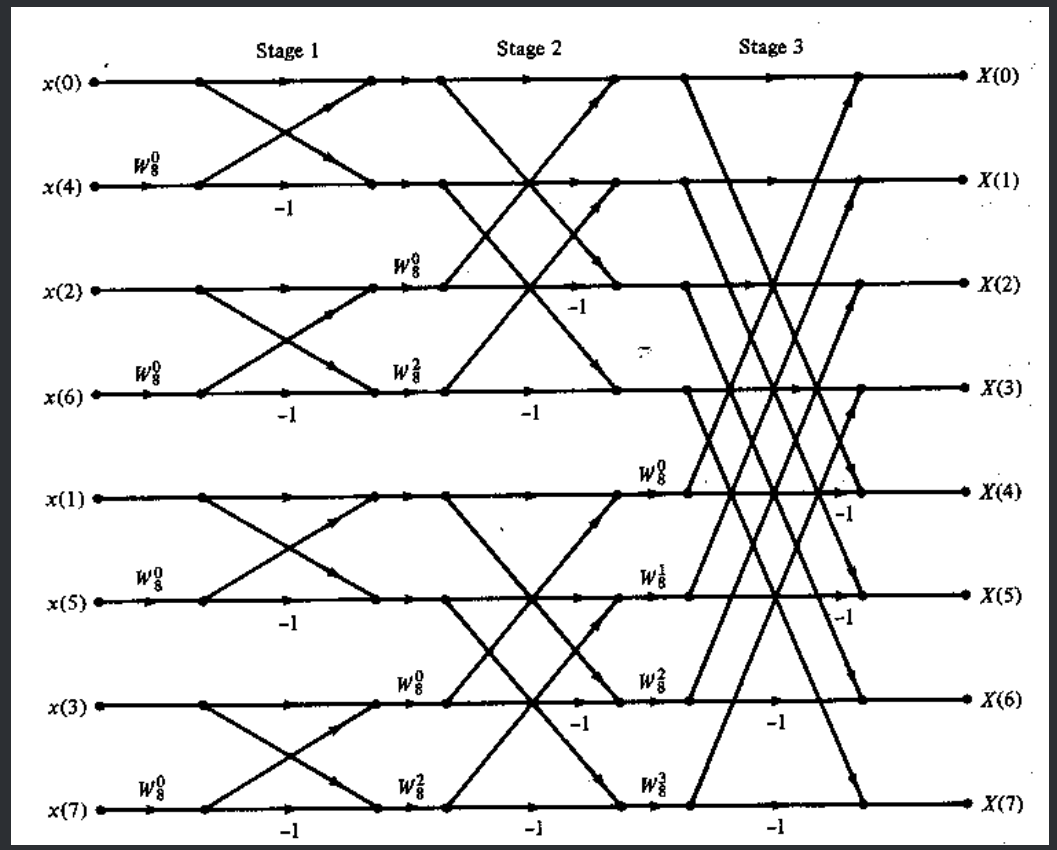
Typické výpočetně náročné operace

- NTT (Number Theoretic Transform),
- násobení matic a vektorů obsahujících polynomy,
- ostatní iterativní části algoritmů (podpisy)
  - vstupy následujících iterací NEZÁVISLÉ na výstupu předchozích iterací

- vstupy následujících iterací NEZÁVISLÉ na výstupu předchozích iterací

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Transformace NTT

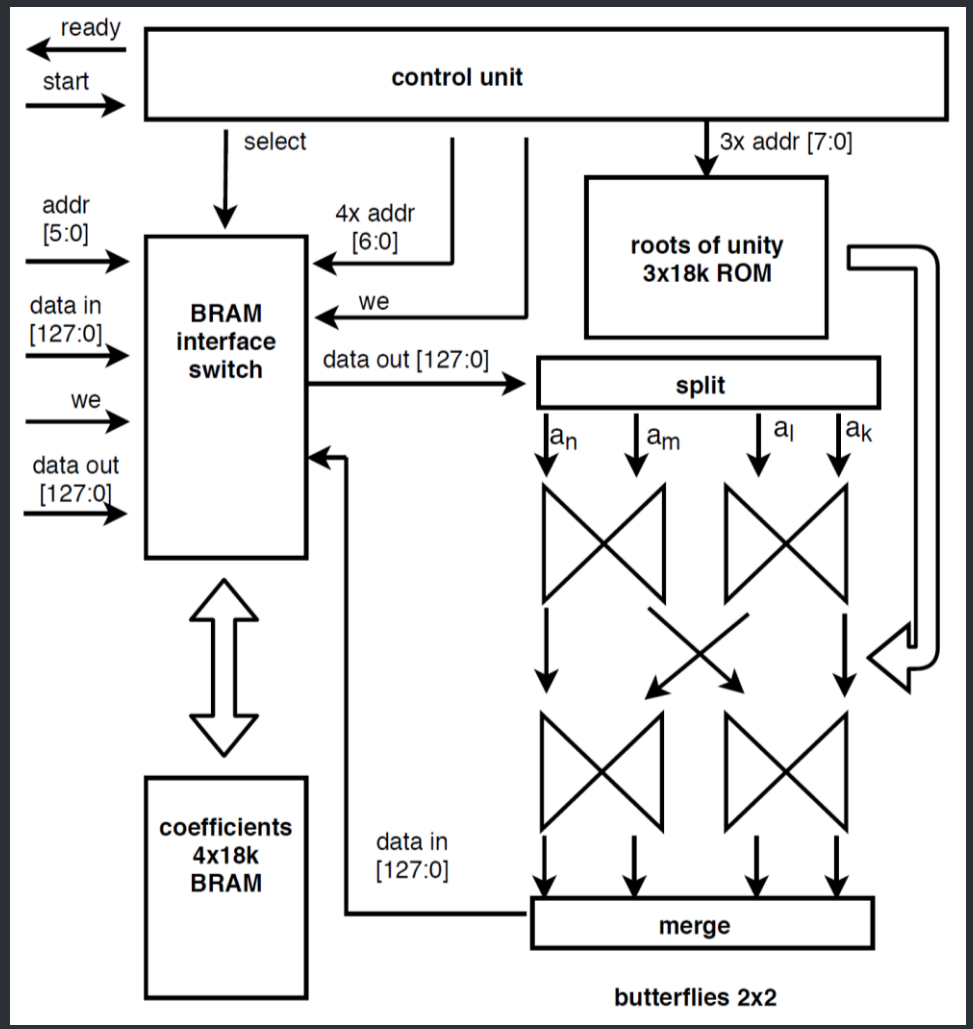


Algoritmus vycházející z FFT

- Pracuje v modulární aritmetice, nikoliv v oboru komplexních čísel,
- výpočet v iteracích,
- transformace 256 koeficientů v 8 iteracích ($2^8 = 256$)
- budoucí iterace závislé na předchozích iteracích ☹

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Implementace NTT



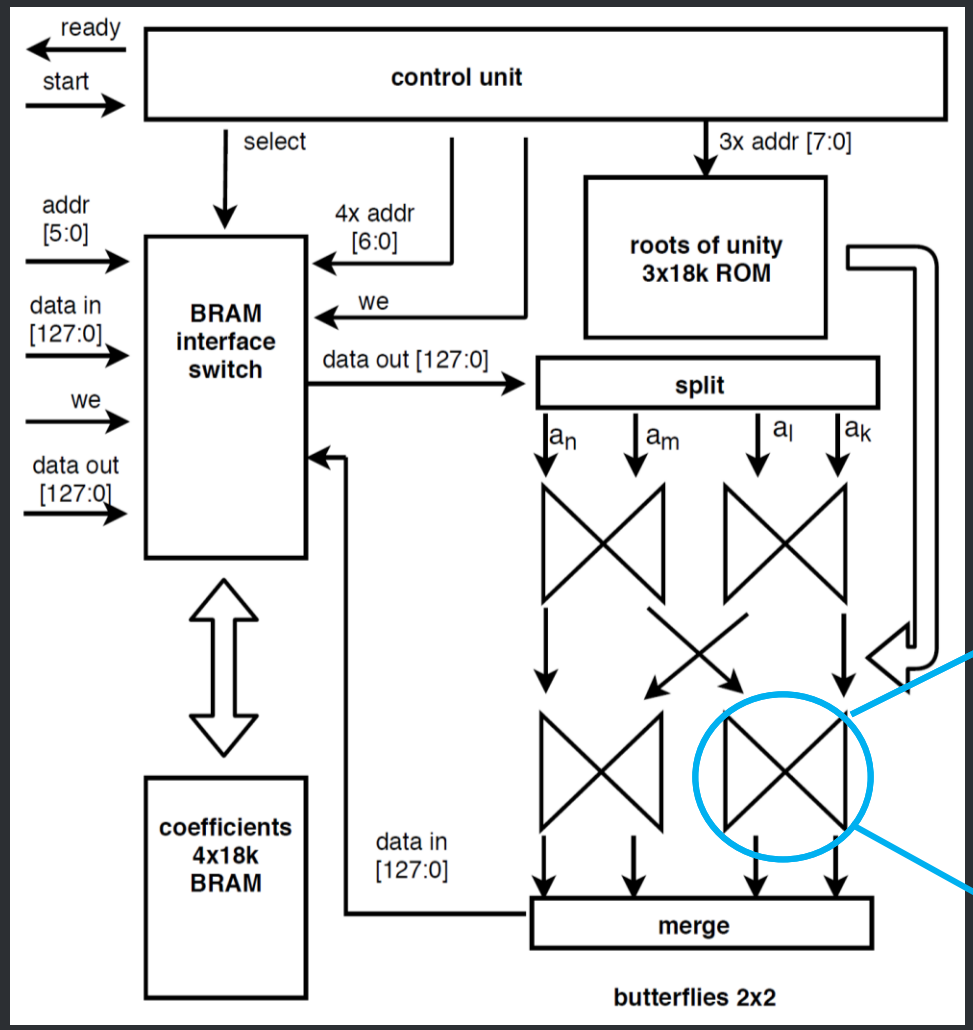
Paralelizace pomocí 4 motýlků

- Motýlci v rozložení 2x2,
- paralelní výpočet 2 iterací,
- počet iterací snížen z 8 na 4,
- minimální počet motýlků pro paralelizaci *n* iterací:

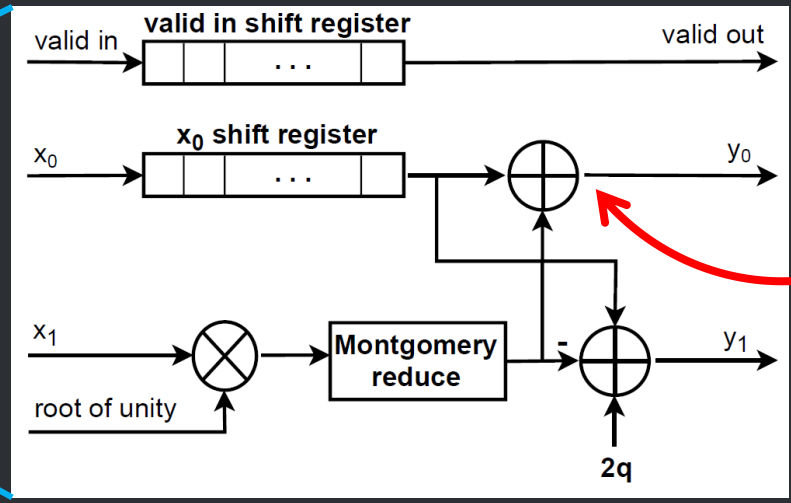
  *n*2^(n-1),*
- taktovací frekvence 632 MHz.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Implementace NTT



	Utilization	
	NTT	INTT
LUT	1798	2547
FF	2532	3889
DSP48	48	84
LUTRAM	438	762
BRAM	3,5	3,5



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Implementace násobení matic

$$As + e = t$$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} + \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{bmatrix}$$

$$p_0x^0 + p_1x^1 + \ldots + p_{255}x^{255}$$

Vhodný algoritmus pro FPGA

- Paralelní výpočet jednotlivých řádků,
- paralelní výpočet NTT,
- v rámci každého řádku paralelní násobení polynomů v NTT doméně,
- paralelní výpočet inverzní NTT

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Implementace Dilithia

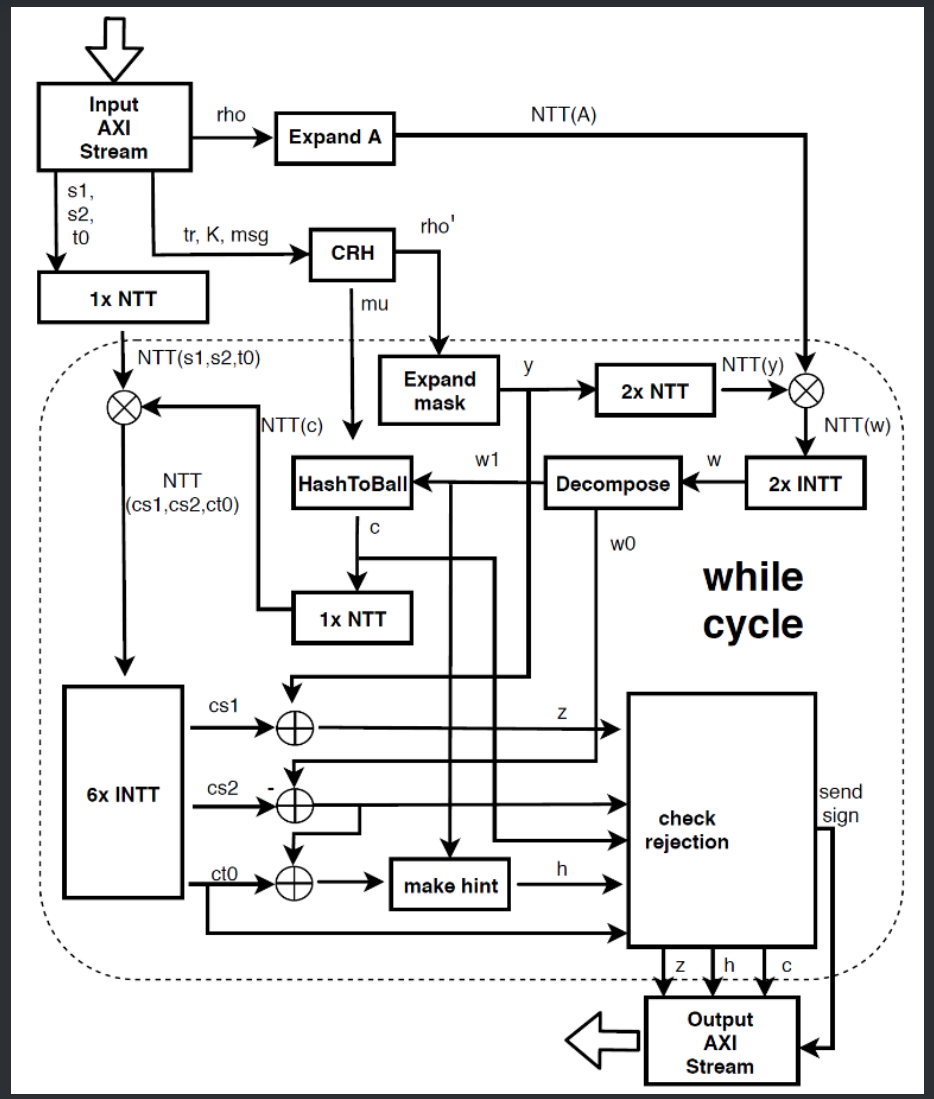
Podpisová část:

Sign(sk, M)

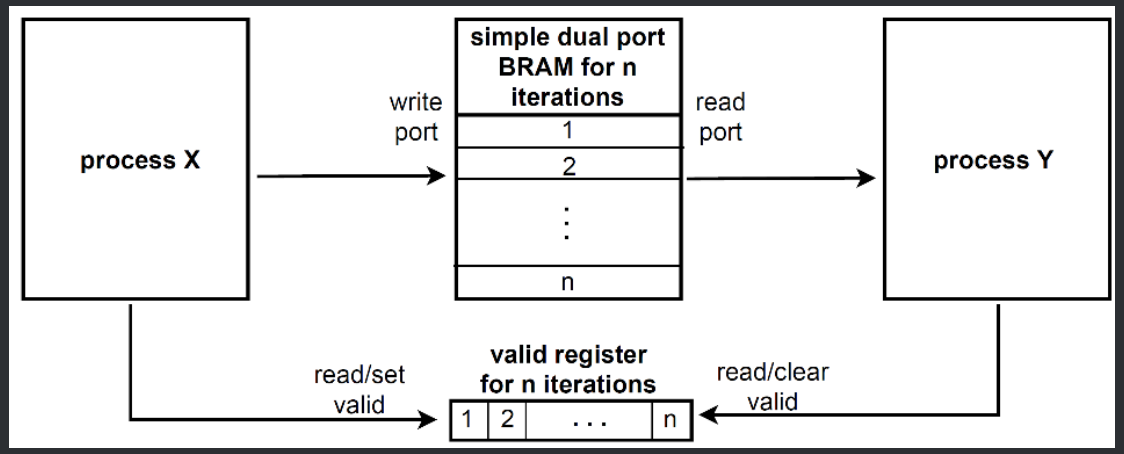
09 $\mathbf{A} \in R_q^{k \times \ell} := \mathsf{ExpandA}(\rho)$ ▷ $\mathbf{A}$ is generated and stored in NTT Representation as $\hat{\mathbf{A}}$
10 $\mu \in \{0,1\}^{512} := \mathrm{H}(tr \parallel M)$
11 $\kappa := 0$, $(\mathbf{z}, \mathbf{h}) := \perp$
12 $\rho' \in \{0,1\}^{512} := \mathrm{H}(K \parallel \mu)$ (or $\rho' \leftarrow \{0,1\}^{512}$ for randomized signing)
13 **while** $(\mathbf{z}, \mathbf{h}) = \perp$ do ▷ Pre-compute $\hat{\mathbf{s}}_1 := \mathtt{NTT}(\mathbf{s}_1)$, $\hat{\mathbf{s}}_2 := \mathtt{NTT}(\mathbf{s}_2)$, and $\hat{\mathbf{t}}_0 := \mathtt{NTT}(\mathbf{t}_0)$
14 $\quad \mathbf{y} \in \tilde{S}_{\gamma_1}^{\ell} := \mathsf{ExpandMask}(\rho', \kappa)$
15 $\quad \mathbf{w} := \mathbf{A}\mathbf{y}$ ▷ $\mathbf{w} := \mathtt{NTT}^{-1}(\hat{\mathbf{A}} \cdot \mathtt{NTT}(\mathbf{y}))$
16 $\quad \mathbf{w}_1 := \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$
17 $\quad \tilde{c} \in \{0,1\}^{256} := \mathrm{H}(\mu \parallel \mathbf{w}_1)$
18 $\quad c \in B_\tau := \mathsf{SampleInBall}(\tilde{c})$ ▷ Store c in NTT representation as $\hat{c} = \mathtt{NTT}(c)$
19 $\quad \mathbf{z} := \mathbf{y} + c\mathbf{s}_1$ ▷ Compute $c\mathbf{s}_1$ as $\mathtt{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}}_1)$
20 $\quad \mathbf{r}_0 := \mathsf{LowBits}_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$ ▷ Compute $c\mathbf{s}_2$ as $\mathtt{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{s}}_2)$
21 $\quad$ **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$, then $(\mathbf{z}, \mathbf{h}) := \perp$
22 $\quad$ **else**
23 $\quad\quad \mathbf{h} := \mathsf{MakeHint}_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$ ▷ Compute $c\mathbf{t}_0$ as $\mathtt{NTT}^{-1}(\hat{c} \cdot \hat{\mathbf{t}}_0)$
24 $\quad\quad$ **if** $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ **or** the # of 1's in $\mathbf{h}$ is greater than ω, then $(\mathbf{z}, \mathbf{h}) := \perp$
25 $\quad \kappa := \kappa + \ell$
26 **return** $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Implementace Dilithia

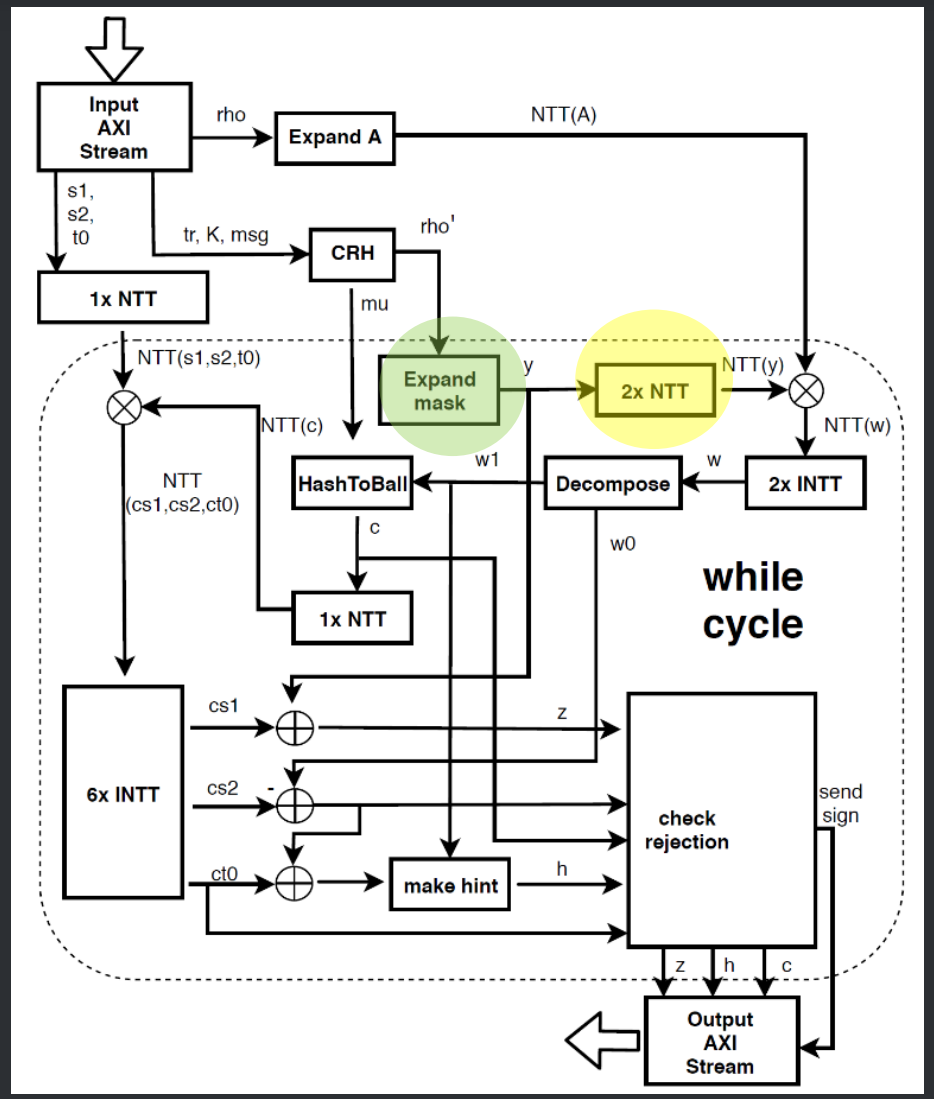


- 18 paralelních procesů,
- 12 paralelních NTT/INTT blocků,
- paralelizace while smyčky.

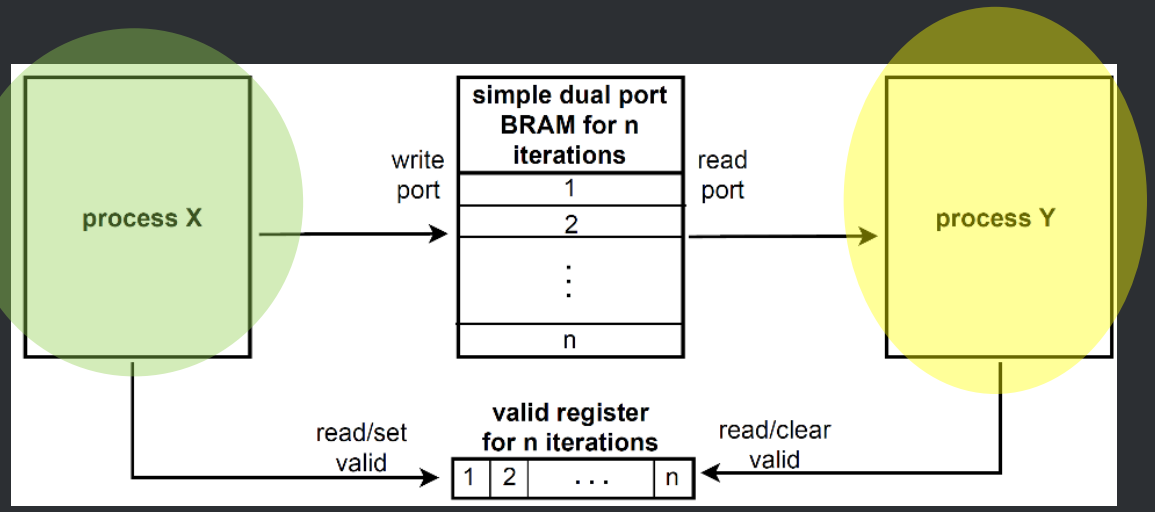


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Implementace Dilithia

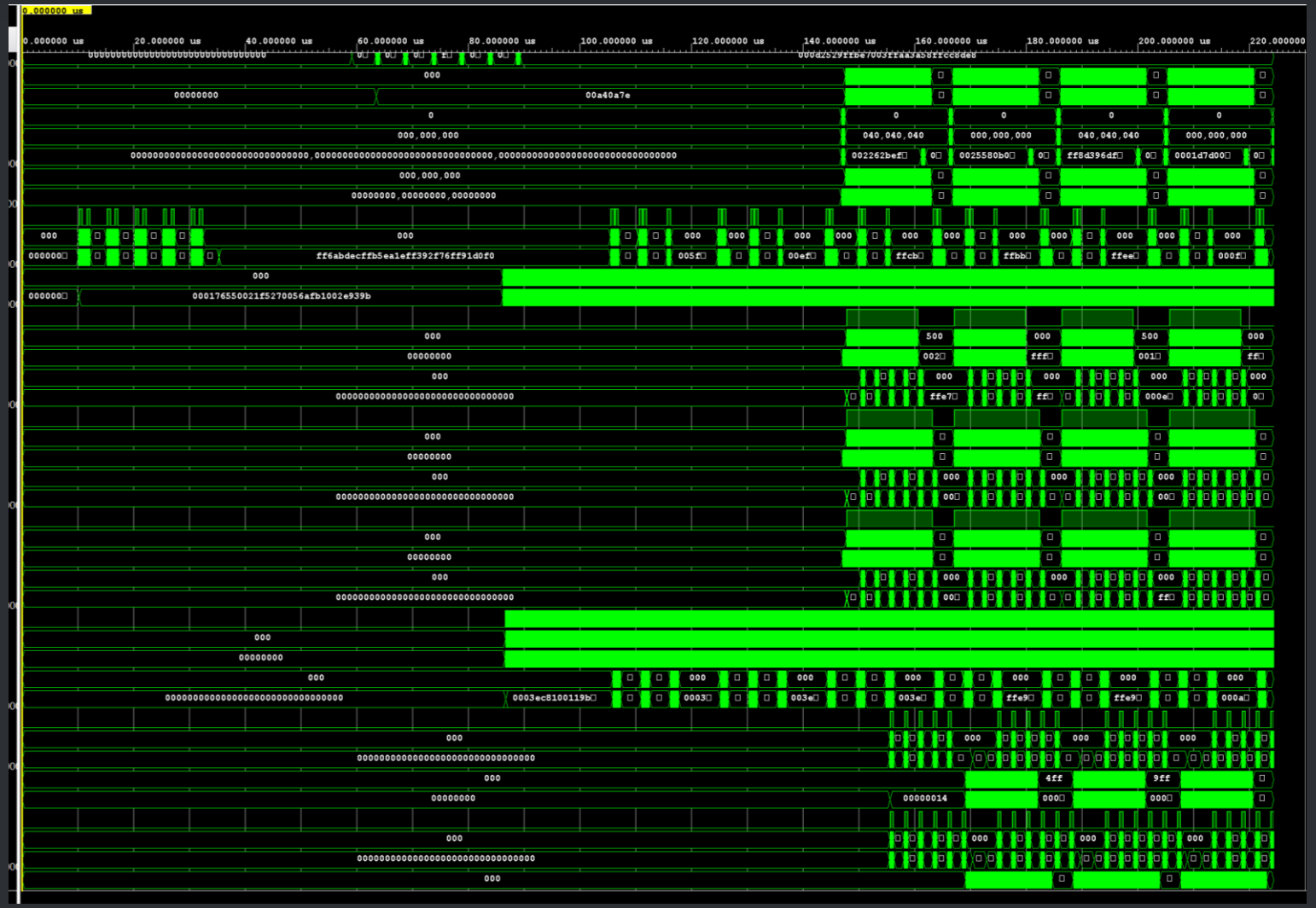


- 18 paralelních procesů,
- 12 paralelních NTT/INTT blocků,
- paralelizace while smyčky.



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

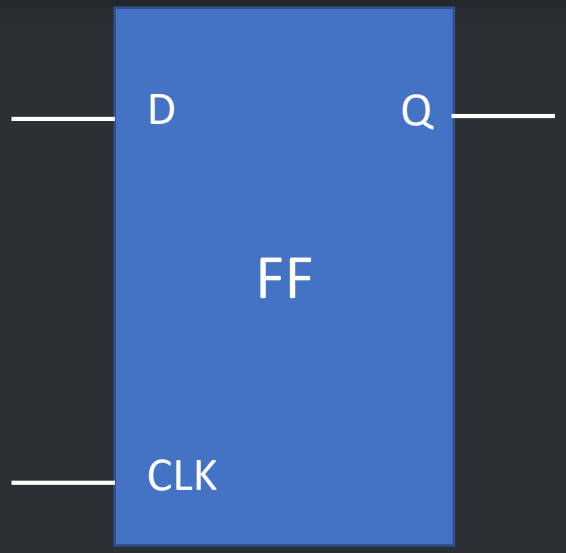
Verifikace designu



- Hlavní metodou jsou funkční simulace, které odhalí většinu chyb.
- Trvají podstatně kratší dobu než syntéza a implementace na FPGA.

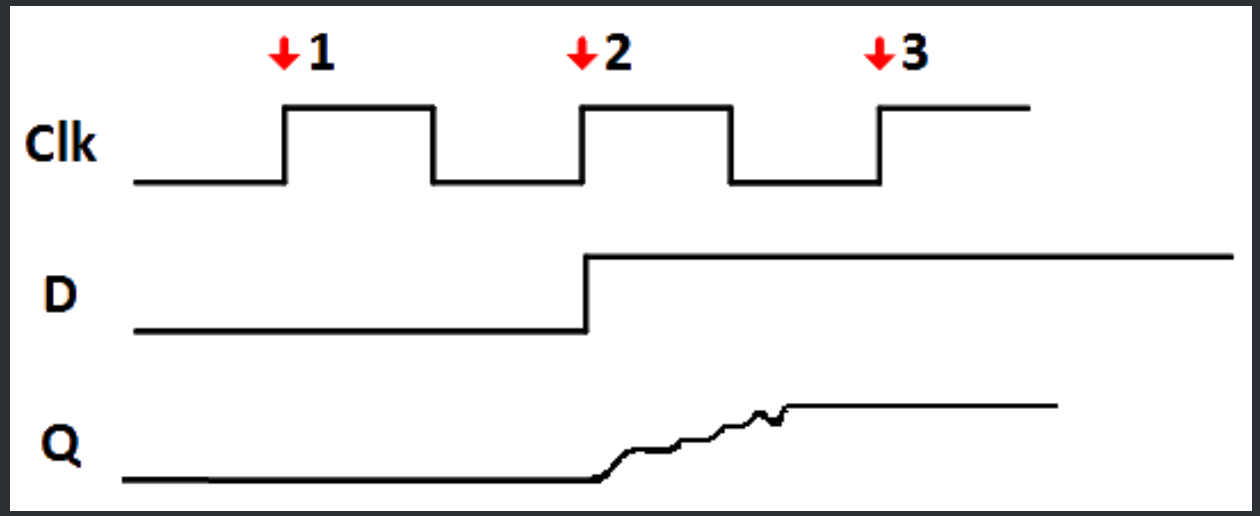
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Verifikace designu



Když se výsledky simulace liší s realitou

- Metastabilita,



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Verifikace designu

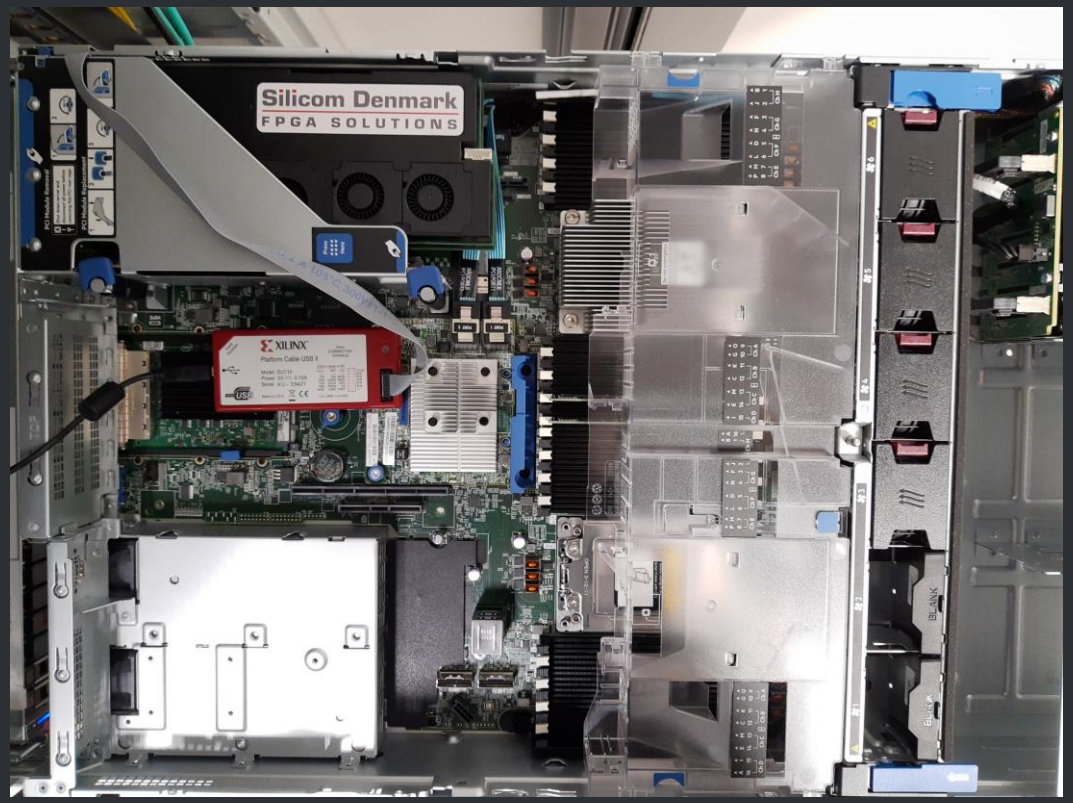
```
…
signal x : integer range 0 to 15;
…
begin
     …
     if x = 16 then
          …
     end if;
…
```

Když se výsledky simulace liší s realitou

- Metastabilita,
- Optimalizace designu během implementace,
- …

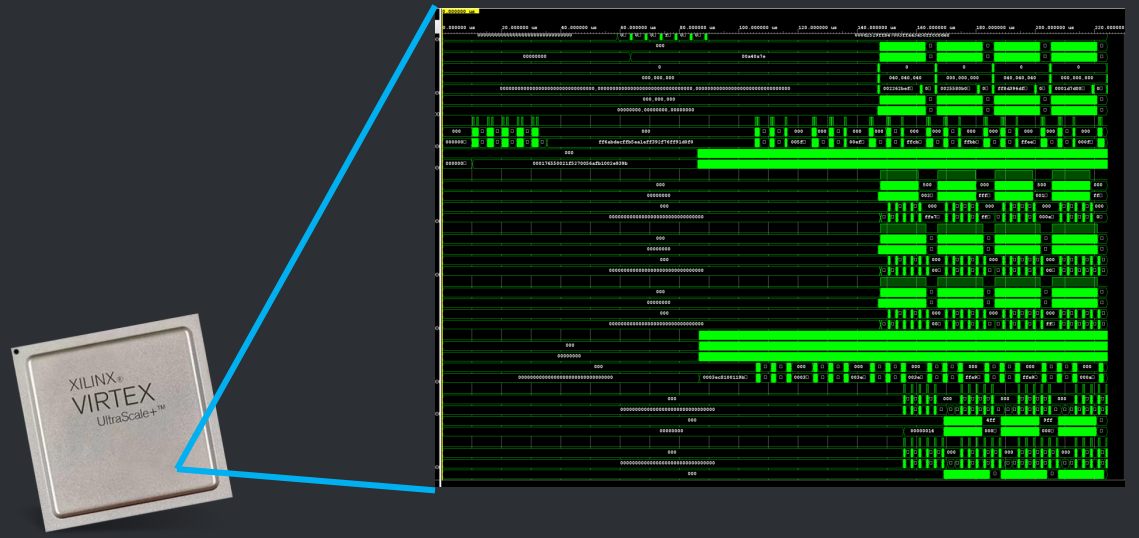
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Ladění designu na FPGA



ILA (Integrated Logic Analyzer)

- Logické analyzátory integrované přímo v FPGA,
- odhalí většinu záludných chyb,
- spotřebovávají hardwarové zdroje (BRAM, …),
- nová konfigurace sond = nová syntéza.



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Děkuji Vám za pozornost ☺