# Využití grafických procesorů pro dekódování LDPC kódů

Jan Broulím
Katedra aplikované elektroniky a telekomunikací
Fakulta elektrotechnická
Západočeská univerzita v Plzni
broulim@kae.zcu.cz

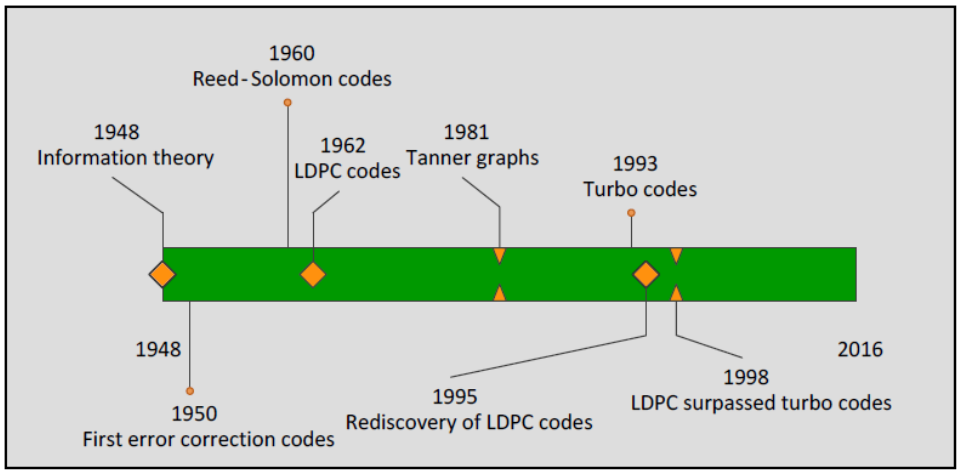
# Utilization of graphics processing units for implementing the LDPC decoder

*Abstract* – **In this paper, the utilization of Graphics Processing Units for parallel LDPC decoding is proposed. The proposed algorithm allows decoding of any irregular LDPC code. The principle of the parallel algorithm is briefly presented, and the performance comparison against serial decoding with the use of Central Processing Units is given in this paper.**

*Keywords – LDPC; GPU; OpenCL; CUDA; parallel algorithms; Belief Propagation; error correction*

## I. Úvod

Grafické procesorové jednotky (Graphics Processing Units, GPUs) představují v současné době perspektivní platformu pro realizaci časově náročných výpočtů. S využitím dostupných frameworků (zejména OpenCL [6] a CUDA [7]) umožňují implementaci paralelních algoritmů. Zvláště efektivní se jeví při výpočtech nevyžadujících příliš komplikované operace, ale umožňujících masivní paralelizaci. Sériová realizace je v těchto případech časově náročná a zrychlení při implementaci na GPU může být i více než stonásobné v porovnání s procesorovými jednotkami (CPU).




***Obrázek I.*** **Vybrané milníky v teorii informace**

Low Density Parity Check (LDPC) kódy [2], dekódované pomocí iterativních algoritmů, nabízejí silný potenciál pro paralelizaci výpočtů prováděných dekodérem. V tomto článku je představeno mapování všeobecně využívaného Belief Propagation algoritmu [3] na platformu grafických procesorů, přičemž je kladen důraz na univerzálnost a použitelnost pro dekódování celé množiny binárních iregulárních LDPC kódů (paralelizace tak není limitována např. maximální hodností uzlu v příslušném Tannerově grafu).

## II. PRINCIP PARALELIZACE BELIEF PROPAGATION ALGORITMU

Definujme:

- Vzestupně seřazenou n-tici variable uzlů $\mathbf{v} = (v_j)$ a příslušnou n-tici $\mathbf{c} = (c_i)$ tak, že $(c_i, v_j)$ představuje hranu v Tannerově grafu,
- N-tici indexů hran $\mathbf{e} = (0,1,2, \dots, |\mathbf{c}|)$,
- N-tici připojených hran $\mathbf{t} = (t_k)$ k danému variable uzlu $v_k$; $t_k = |v_k|, v_k \in \mathbf{v}$,
- N-tici pozic $\mathbf{s} = (s_k)$ pro začátek iterace pro výpočet zprávy předávané po hraně $e_k$; $s_k = \mathrm{argmin}_k(v_k : v_k \in \mathbf{v})$,
- N-tici relativních pozic hran $\mathbf{u} = (u_k)$; $u_k = k - |(v_q) : q < k, v_q \neq v_k|$

Obdobně můžeme definovat n-tice pro druhou polovinu iterace, s uvážením seřazené n-tice indexů check uzlů. Takové n-tice jsou pak v popisu algoritmu označeny s pruhem. Paralelizaci iterativního předávání zpráv bez komplikovaných výpočtů s indexy polí ukazuje Algoritmus 1. Parametr *lgsize* počet synchronizovatelných vláken.

ALGORITMUS 1 PARALELNÍ PŘEDÁVÁNÍ ZPRÁV

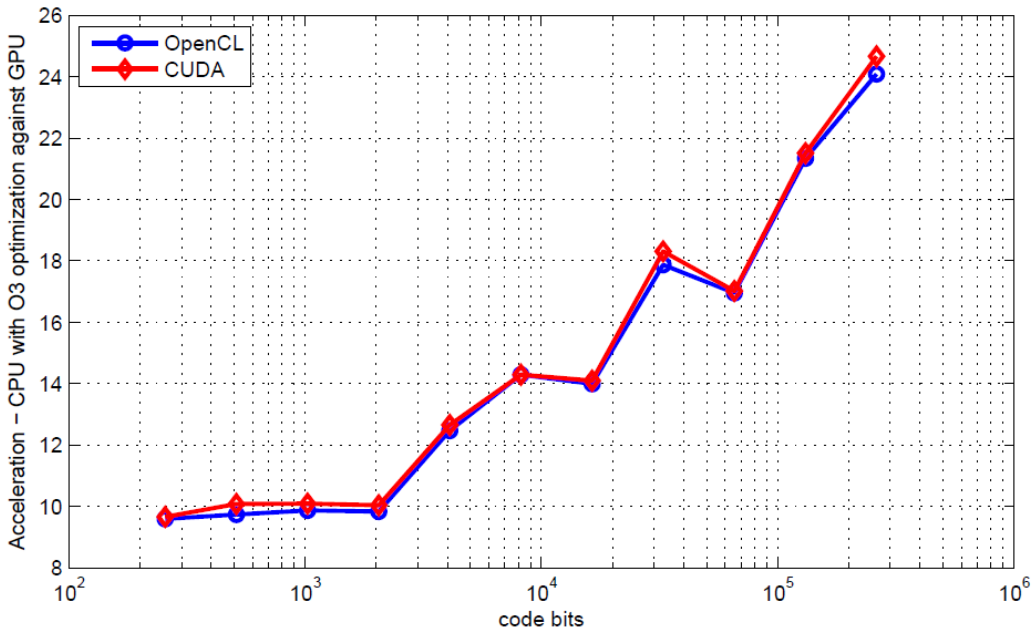
```
1:  procedure ITERATE TO CHECK NODES
    ▷ Half on an iteration
    Input: r – incoming values e, s, t, u
    Output: q
2:      for (p = 0; p < totaledges; p+ = lgsize) do
3:          for i = s_{lid+p} to s_{lid+p} + t_{lid+p} − 1 do
4:              if i = u_{lid+p} + s_{lid+p} then continue
5:              end if
6:              value = perform calculations
7:          end for
8:          index = e_{lid+p}
9:          q_{index} = value
10:     end for
11: end procedure

12: procedure ITERATE TO VARIABLE NODES
    ▷ Half on an iteration
    Input: q – incoming values ē, s̄, t̄, ū
    Output: r
13:     for (p = 0; p < totaledges; p+ = lgsize) do
14:         for i = s̄_{lid+p} to s̄_{lid+p} + t̄_{lid+p} − 1 do
15:             if i = ū_{lid+p} + s̄_{lid+p} then continue
16:             end if
17:             value = perform calculations
18:         end for
19:         index = ē_{lid+p}
20:         r_{index} = value
21:     end for
22: end procedure
```

ALGORITMUS 2 VÝPOČET ODHADU

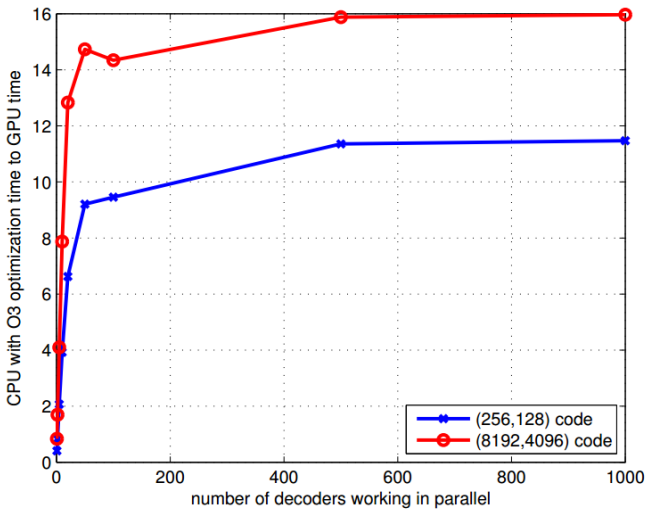
```
1: procedure CALCULATE ESTIMATION
   ▷ Parallel approach
   Input: r – incoming values s, t, v
   Output: ĉ
2:     for (p = 0; p < totaledges; p+ = lgsize) do
3:         Q_1 = r_{lid+p}
4:         Q_0 = 1 − r_{lid+p}
5:         for i = s_{lid+p} to s_{lid+p} + t_{lid+p} − 1 do
6:             Q_1 = Q_1 r_{i+p}
7:             Q_0 = Q_0(1 − r_{i+p})
8:         end for
9:         index = v_{lid+p}
10:        if Q_1 > Q_0 then  ĉ_index = 1
11:        else  ĉ_index = 0
12:        end if
13:        index = v_{lid+p}
14:        q_index = value
15:    end for
16: end procedure
```

## III. MĚŘENÍ ČASOVÉ NÁROČNOSTI

Měření proběhlo na platformách NVIDIA Tesla K40 (běžící na frekvenci 745 MHz) a Intel Xeon E5-2695v2 (běžící na frekvenci 2.4 GHz). Obrázek II ukazuje porovnání časové náročnost dekódování při $E_B/N_0 = 2$dB. Vybrán byl kód dle standardu NASA CCSDS, přičemž delší kódy byly generovány pomocí protografické expanze. Z grafu je patrná zejména zvyšující se efektivita představeného algoritmu s délkou kódu.




***Obrázek II.*** **Porovnání akcelerace GPU vs CPU**




***Obrázek III.*** **Závislost akcelerace na počtu dekodérů pracujících paralelně**

## IV. ZÁVĚR

Využití grafických procesorových jednotek pro implementování časově náročných algoritmů je účinné zejména při vhodně realizované paralelizaci Představená paralelizace Belief Propagation dekodéru se při měření ukázala cca 25× rychlejší pro testované délky kódů než stejný algoritmus sériový algoritmus běžící na CPU (uvažována O3 optimalizace zdrojového kódu). S vynecháním O3 optimalizace je zrychlení až 70 násobné. Zmiňovaná problematika je podrobněji diskutována v připravovaném článku [1].

## PODĚKOVÁNÍ

Tento článek vznikl za podpory interního projektu na podporu studentských vědeckých konferencí SVK-2016-006 a projektu SGS-2015-002: Moderní metody řešení, návrh a aplikace elektronických a komunikačních systémů.

## LITERATURA

[1] J. BROULIM, A. AYRIYAN, V. GEORGIEV, „OpenCL/CUDA algorithms for parallel decoding of any irregular LDPC code using GPU“. , [Online]. 2016. [cit. 2016-09-25]. Dostupné z: http://arxiv.org/abs/1609.01567.

[2] G. GALLAGER, Low Density Parity Check Codes, Transactions of the IRE Professional Group on Information Theory, Vol. IT-8, January 1962, pp. 2l-28.

[3] N. WIBERG, Codes and Decoding on General Graphs. PhD thesis, Dept. of Electrical Engineering, Lionkoing, Sweden, 1996. Lionkoing studies in Science and Technologz. Dissertation No. 440.

[4] Implementing FPGA Design with the OpenCL Standard, Altera, 2013.

[5] Short Block Length LDPC Codes for TC Synchronization and Channel Coding. CCSDS Experimental Specification. NASA, 2015.

[6] Khronos OpenCLWorking Group, The OpenCL Specification, [Online]. 2011. [cit. 2016-09-23]. Dostupné z: https://www.khronos.org/registry/cl/specs/opencl-1.2.pdf.

[7] NVIDIA Corporation, Cuda Runtime API, Reference manual, , [Online]. 2015. [cit. 2016-09-23]. Dostupné z: http://docs.nvidia.com/cuda/pdf/CUDA Runtime API.pdf.