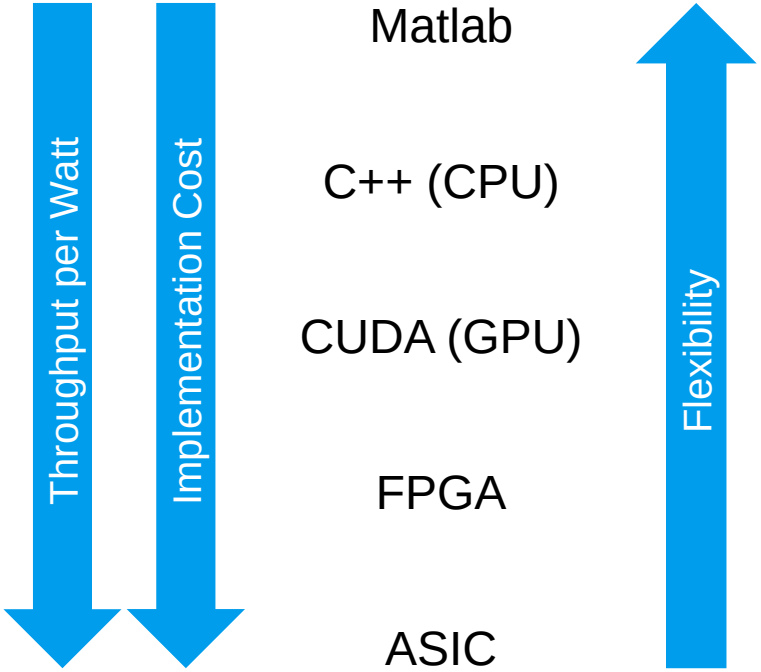
Decoding on GPUs
on the Example of Turbo Product Codes

Stefan Dierks
28.02.2019

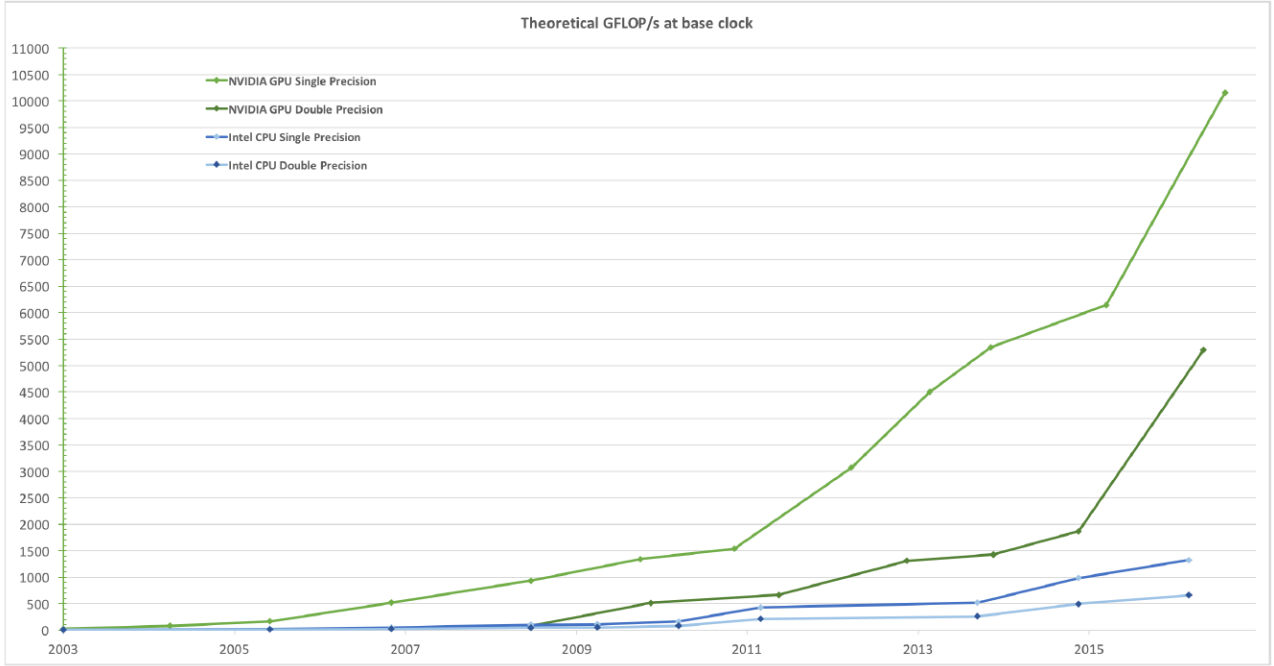
ROHDE&SCHWARZ

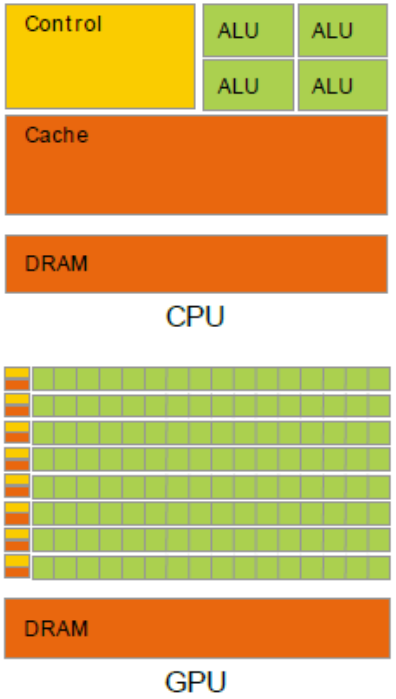
Why do we use GPUs to decode?

- Software defined receiver
- Many different codes (e.g., turbo product codes, turbo convolutional codes, LDPC, BCH codes)
- Many different code parameters
- Other GPU receiver modules
- Quick development
- Parallel data structures
- Latency is not critical
- Throughput is important
- Small quantity

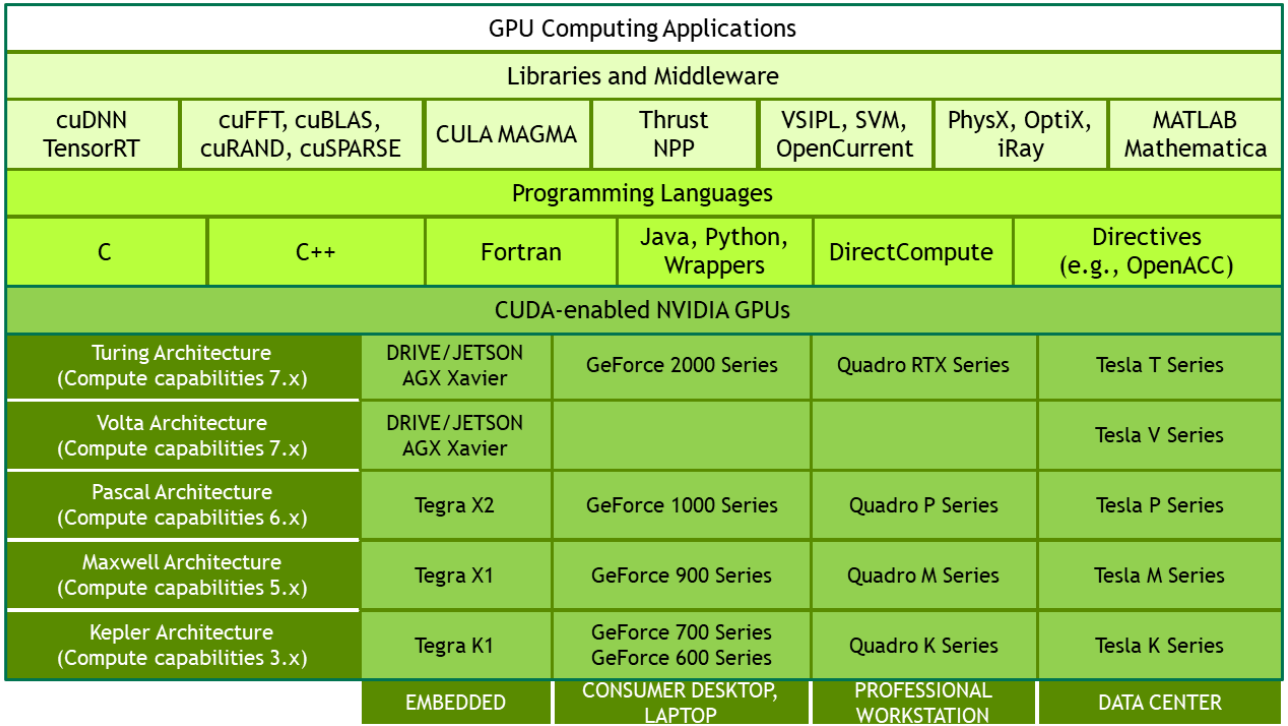


GPU: From Graphics Processing to General Purpose Parallel Computing





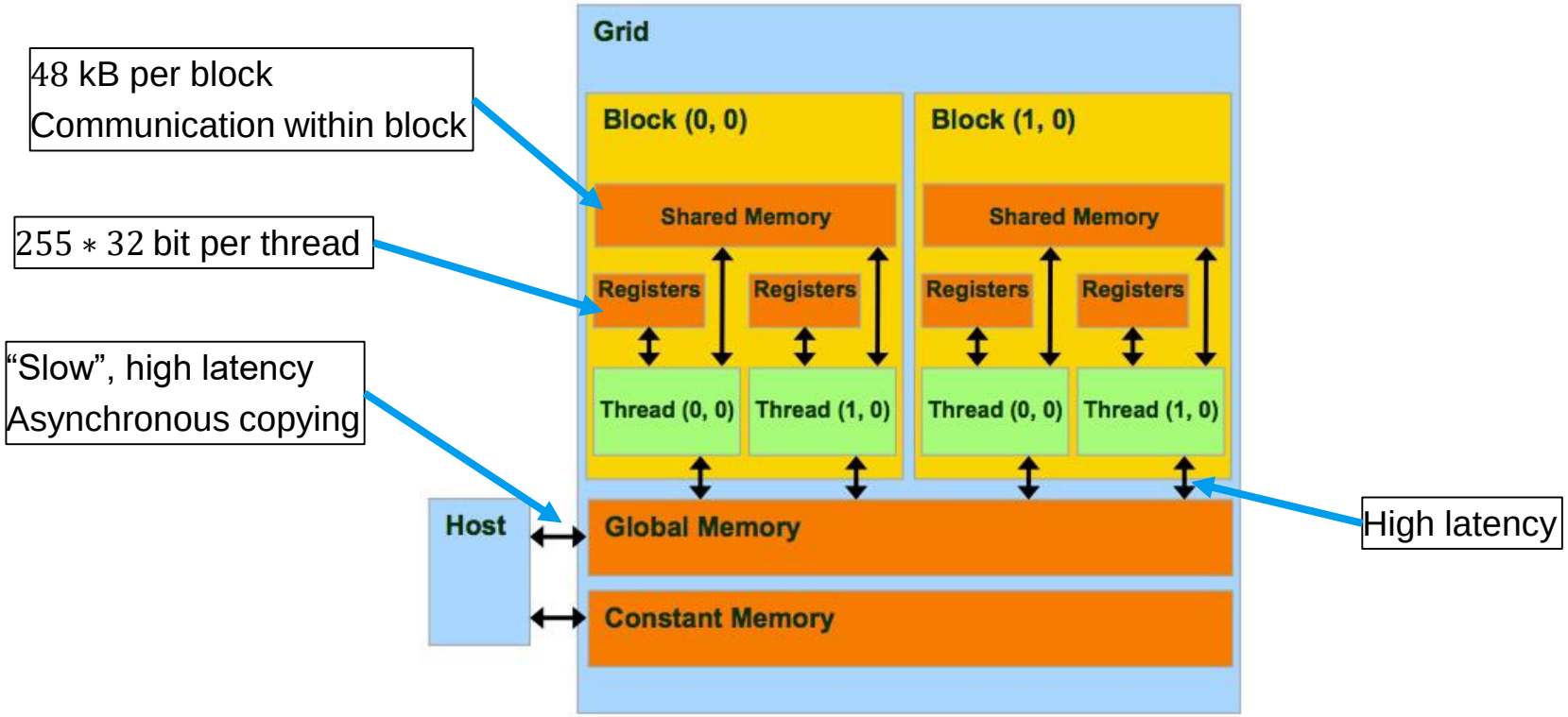
CUDA (Compute Unified Device Architecture)



NVIDIA Tesla V100

Streaming Multiprocessors	80
Cores per SM	2*32
Cores	5120
GPU Boost Clock	1530 MHz
Peak FP32 TFLOPS	15.7
Memory Size	16 GB
TDP	300 Watts
Transistors	21.1 billion
GPU Die Size	815 mm²
Manufacturing Process	12 nm FFN
Cost	≈ 7000€

GPU Memory Hierarchy



Turbo Product Code (TPC)

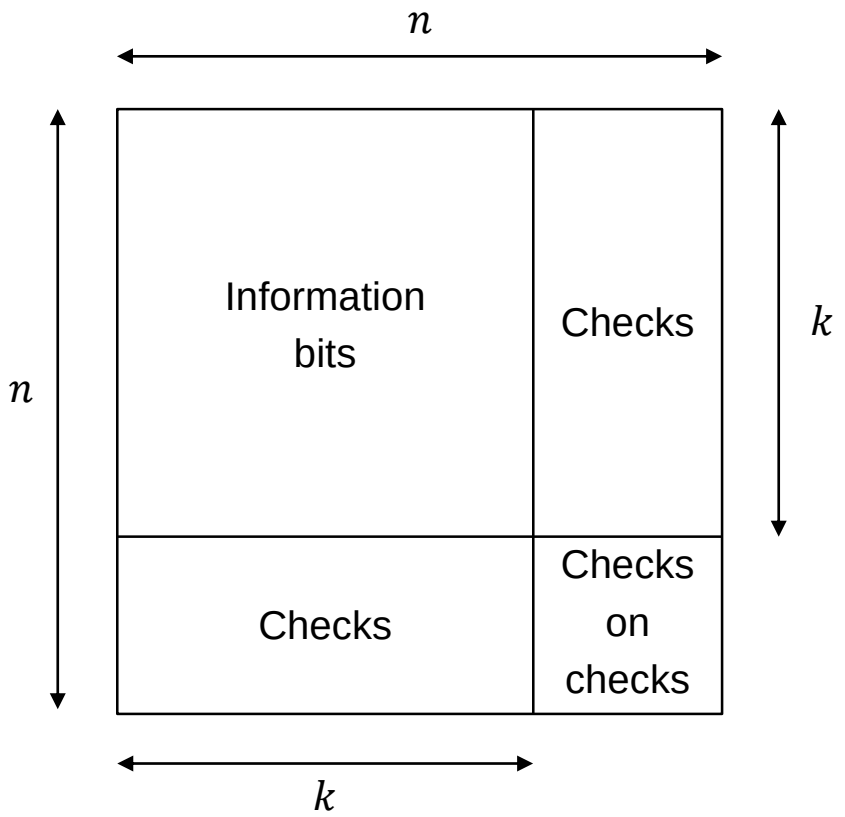
Assume a symmetric TPC:

- Same row and column block code
- Symmetric block-code (n, k, δ)

➢ (n, k, δ) x (n, k, δ) = (n^2, k^2, δ^2) TPC

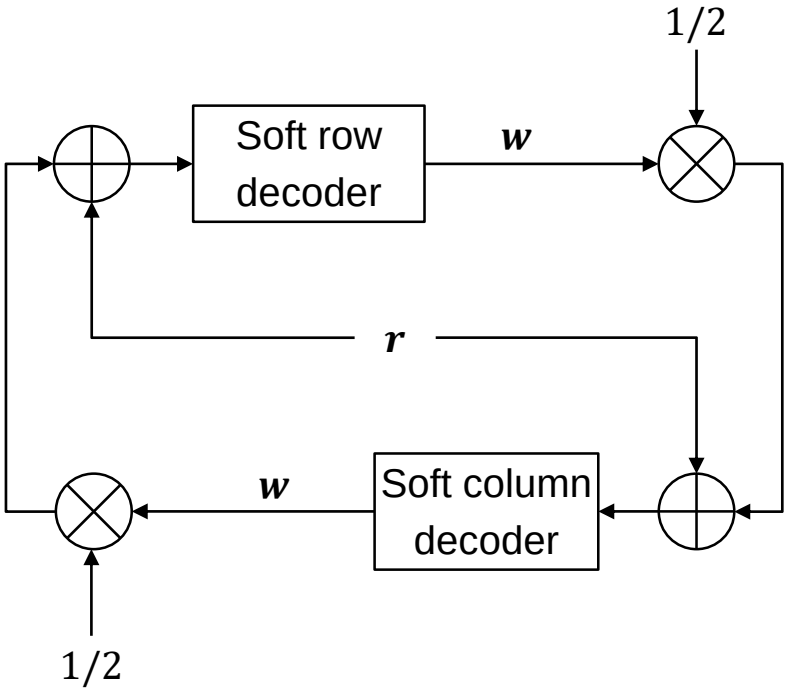
Application

- Satellite communication (e.g., VSAT)
- …



TPC Decoding

- Hard-in hard-out (HiHo) decoder
  - Closed-chain errors remain
- Soft-in soft-out (SiSo) decoder
  - Chase-Pyndiah decoder [Pyndiah, R. M., IEEE Trans. Commun., 1998]
  - “A Low Complexity Decoding Algorithm for Extended Turbo Product Codes” [Xu, C.; et al., IEEE Trans. Wireless Commun., 2008]
  - …
- Hybrid approaches



Low Complexity Soft Block Decoder

1. Input: LLRs $\bar{\boldsymbol{r}} = \boldsymbol{r} + \frac{1}{2}\boldsymbol{w}$
2. ML sequence : $\boldsymbol{y}_j = \begin{cases} 0, \text{where } \bar{\boldsymbol{r}}_j > 0 \\ 1, \text{where } \bar{\boldsymbol{r}}_j \leq 0 \end{cases}$
3. Determine p positions with minimal $|\bar{\boldsymbol{r}}_j|$
4. Generate 2^p test pattern $\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(2^p)}$ (all permutations of p positions)
5. Perturbed sequences: $\boldsymbol{z}^{(i)} = \boldsymbol{y} \oplus \boldsymbol{t}^{(i)}$
6. Hard decode $\boldsymbol{z}^{(i)} \rightarrow \boldsymbol{c}^{(i)}$
7. Analog weight: $d^{(i)} = -\sum(\boldsymbol{c}_j^{(i)} \oplus \boldsymbol{y}_j)|\bar{\boldsymbol{r}}_j|$
8. ML code word: $\boldsymbol{c}^{(\hat{\imath})}$ where $\hat{\imath} = \arg\max_{\hat{\imath}} d^{(i)}$
9. Estimate extrinsic information $\boldsymbol{w}$ of $\boldsymbol{c}^{(\hat{\imath})}$

Estimate extrinsic information of $\boldsymbol{c}_j^{(\hat{\imath})}$:

- Find largest $d^{(i)}$ where $\boldsymbol{c}_j^{(i)} \neq \boldsymbol{c}_j^{(\hat{\imath})}$

$$\boldsymbol{w}_j = \left(d^{(i)} - d^{(\hat{\imath})}\right)\left(2\boldsymbol{c}_j^{(\hat{\imath})} - 1\right) - \bar{\boldsymbol{r}}_j$$

- If $\boldsymbol{c}_j^{(i)} = \boldsymbol{c}_j^{(\hat{\imath})}$ for i ∈ $[1, \ldots, 2^p]$

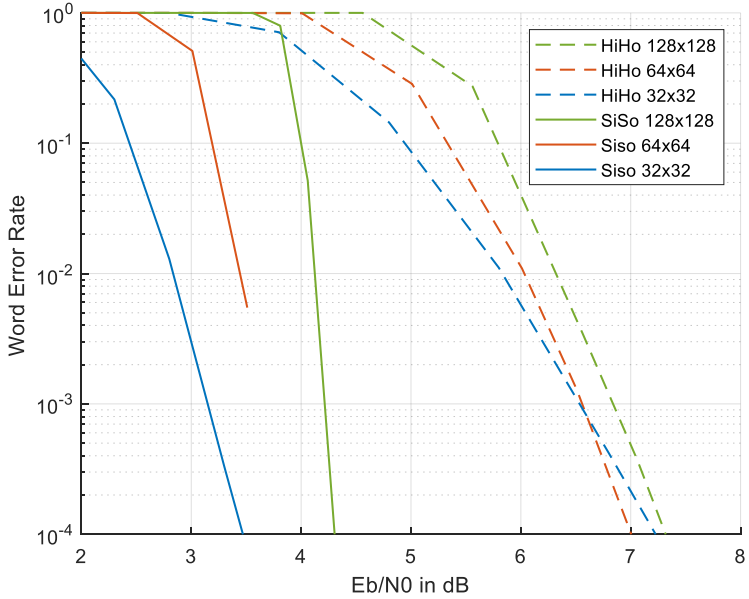
$$\boldsymbol{w}_j = \frac{1}{p}\left(\min_i\left(d^{(i)}\right) - d^{(\hat{\imath})}\right)\left(2\boldsymbol{c}_j^{(\hat{\imath})} - 1\right)$$

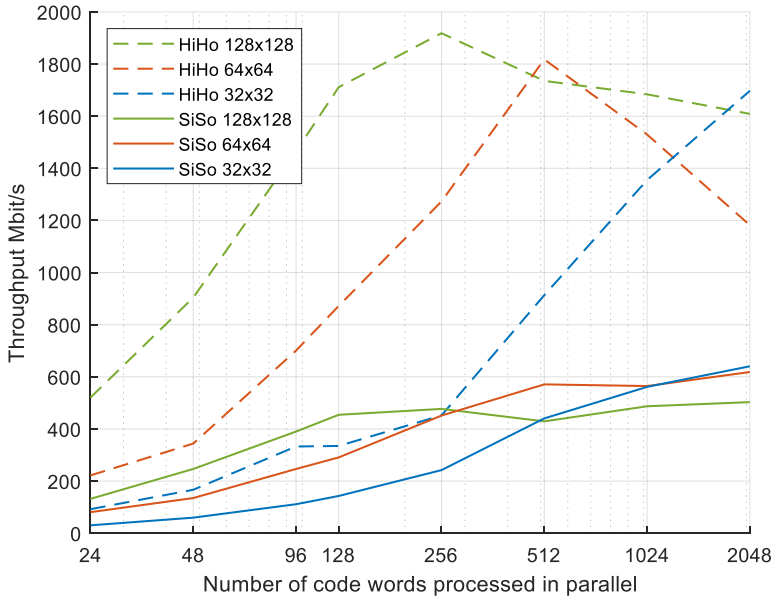
TPC Parameters

- TPC with extended Hamming codes
  - $(32, 26, 4)$ x $(32, 26, 4)$ Code rate: $\frac{676}{1024} = 0.66$ $1 * 32$ threads per block
  - $(64, 57, 4)$ x $(64, 57, 4)$ Code rate: $\frac{3249}{4096} = 0.79$ $2 * 32$ threads per block
  - $(128,120, 4)$ x $(128,120, 4)$ Code rate: $\frac{14400}{16384} = 0.88$ $4 * 32$ threads per block
- TPC extrinsic information matrix fits into shared memory (as 16bit half precision floats)
- Received LLRs fit into registers (as 16bit half precision floats)
- 4 turbo iterations
- HiHo decoder parameters:
  - $4 * 2 * n = 8n$ hard block decoder calls
- SiSo decoder parameters:
  - $p = 3$ bit positions
  - $4 * 2 * 2^p * n = 64n$ hard block decoder calls

- Hard block decoder should be fast:
  - Use hardware intrinsic where possible
  - Provide parameters at compile time

Performance Results





Other GPU Modules

- Turbo convolutional decoder
- LDPC decoder
- BCH decoder
- CRC
- Detector
- Time offset estimator
- Frequency/phase offset estimator
- Resampler