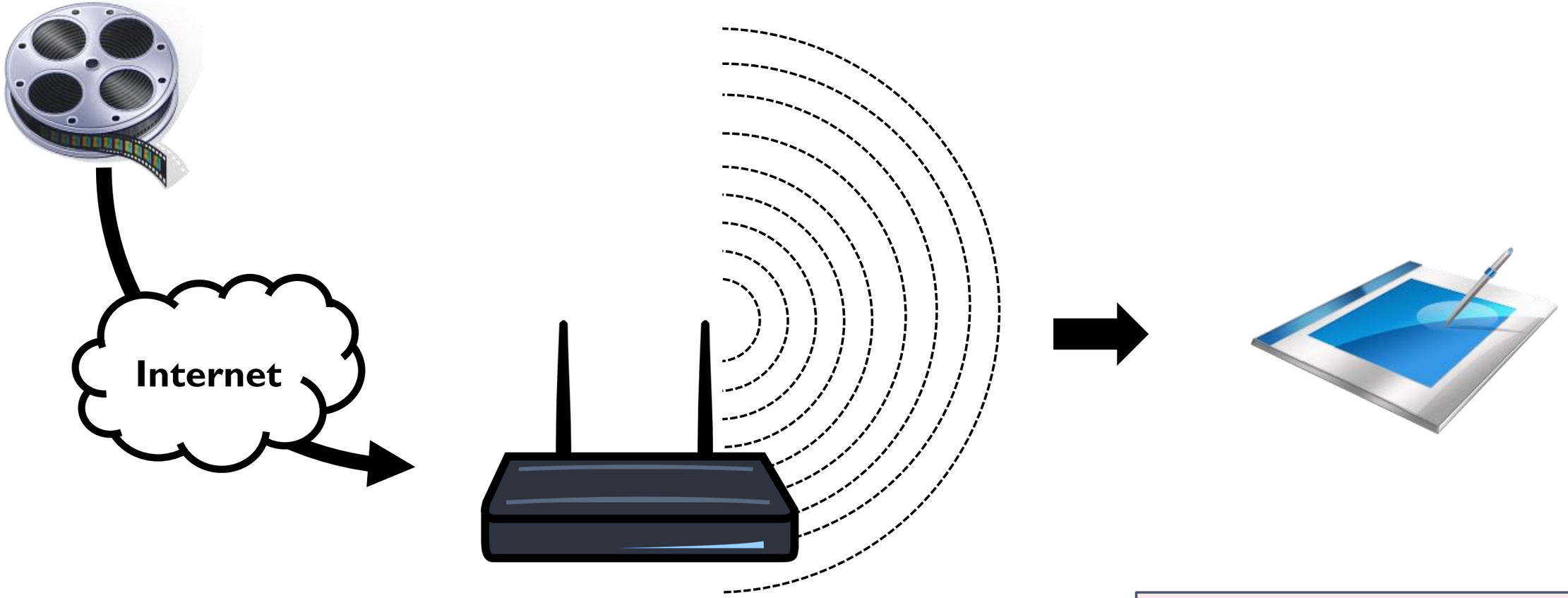
2013 GPU Technology Conference (GTC), March 18-21

Massively Parallel Signal Processing for Wireless Communication Systems

Michael Wu, Guohui Wang, Joseph R. Cavallaro

Department of ECE, Rice University

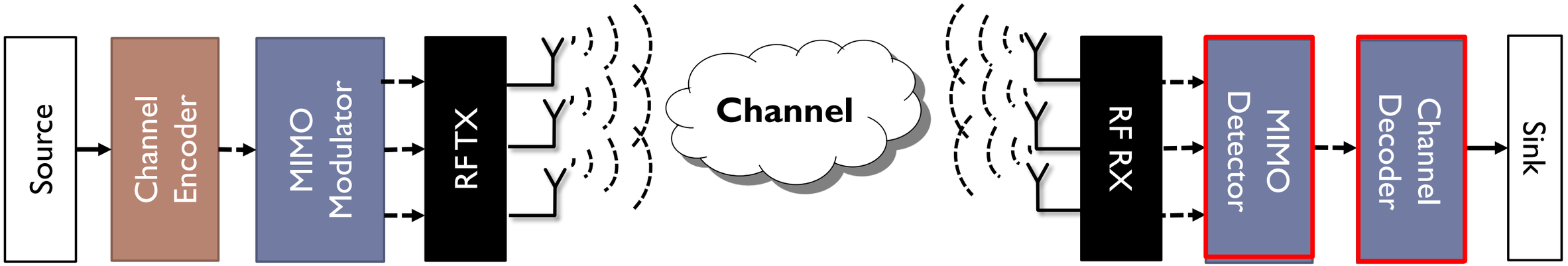
Wireless Communication Systems



Information bits → transmitted signal

Received signal → Information bits

Wireless Communication Systems



- Used in many standards:

- Heavy computations on RX side:
  - MIMO Detector: decouple streams to provide estimates of the tx bits.
  - Channel Decoder: correct errors using redundant data.

Wireless Communication Systems (cont.)

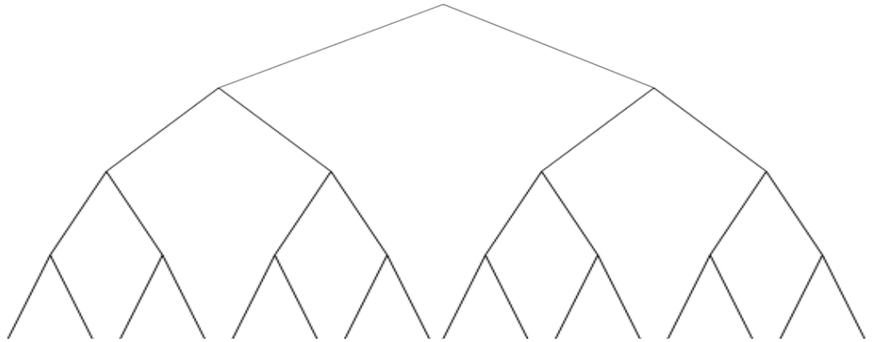
- Related Research Work
  - MIMO detector: *[SAMOS 2010 IEEE-TVT 2012]*
  - Turbo decoder: *[CASES 2010,VLSID 2012]*
  - LDPC decoder: *[ASILOMAR 2008,TPDS 2010, JSPS 2010, SASP 2011]*
  - SDR systems: *[IEEE Comm. Mag 2010, ISRSP 2011]*

- Our previous work
  - **MIMO detector:** *[SiPS 2009,Asilomar 2009, JSPS 2011, SIPS 2012]*
  - Turbo decoder: *[SiPS 2010, JSPS 2011]*
  - **LDPC decoder:** *[SASP 2011,Asilomar 2011]*
  - NB-LDPC decoder: *[Asilomar 2012]*

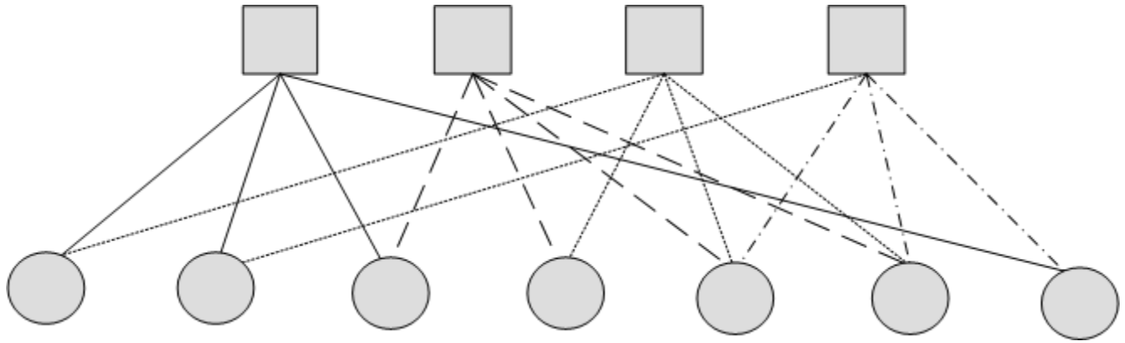
Massively parallel implementations

- Massively parallel implementations:

MIMO Detector



LDPC Decoder



- Tailored algorithms to improve efficiency.
- Results:
  - Achieve high throughput (faster than existing work)
  - Very flexible, can be a good platform for SDR systems.

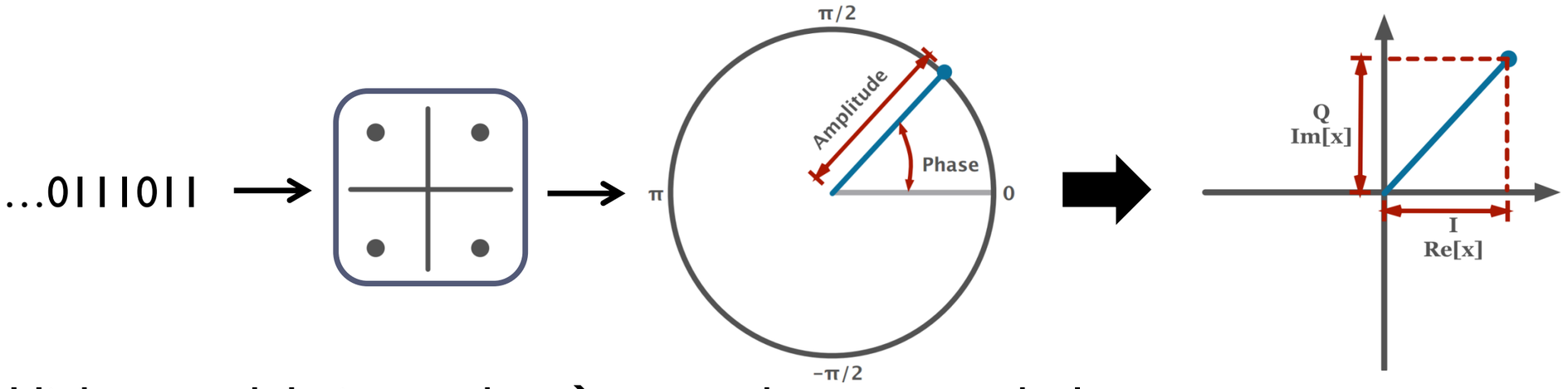
Outline

- MIMO soft detection algorithm on GPU
  - Introduction to MIMO detection
  - Kernel mapping
  - Optimization techniques
  - Experiment results

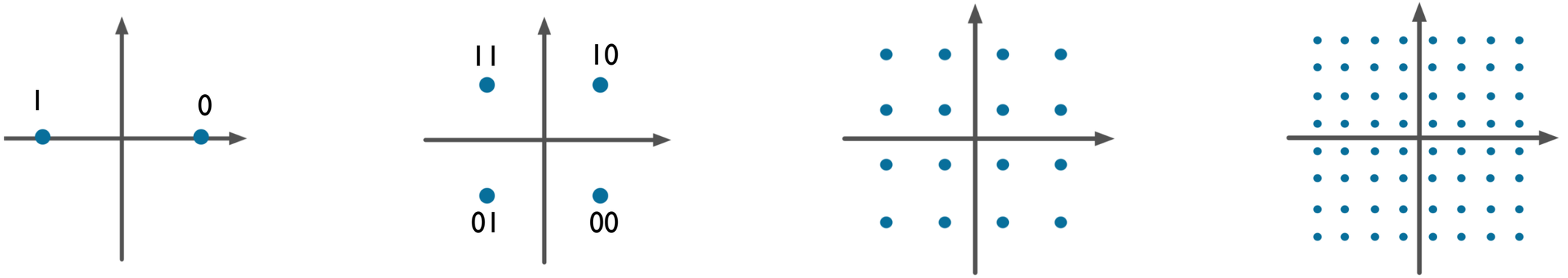
- Multi-standard LDPC Decoder on GPU
  - Introduction to LDPC decoding algorithm
  - Kernel mapping
  - Optimization techniques
  - Experiment results

Modulation

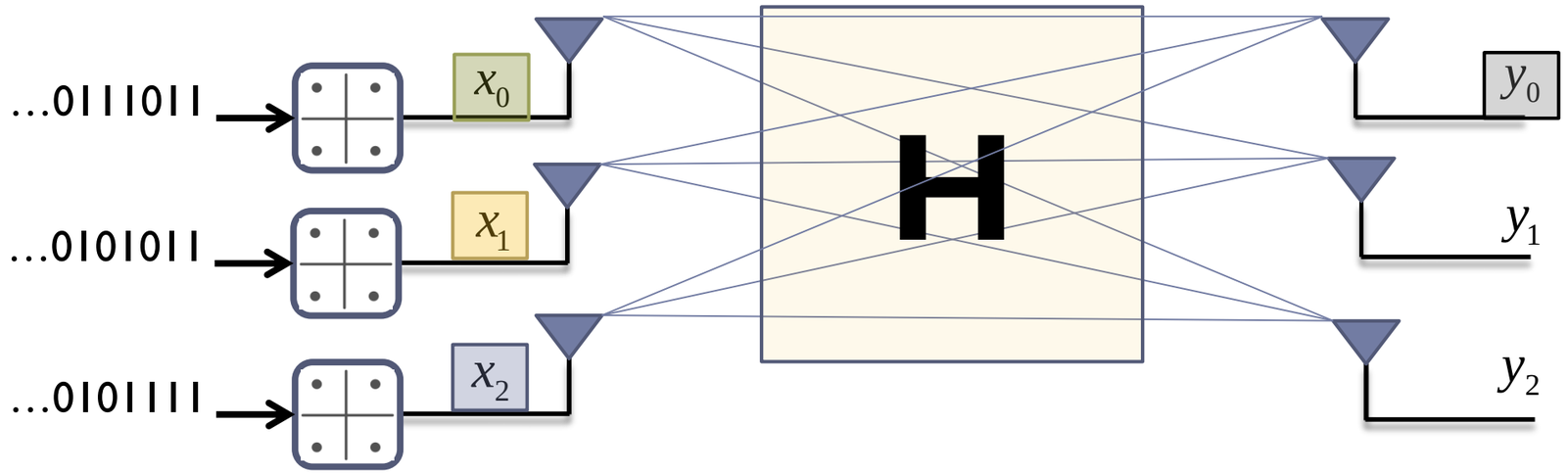
- Encode data in amplitude and phase of a sinusoid



- Higher modulation order → more data per symbol



MIMO System Model

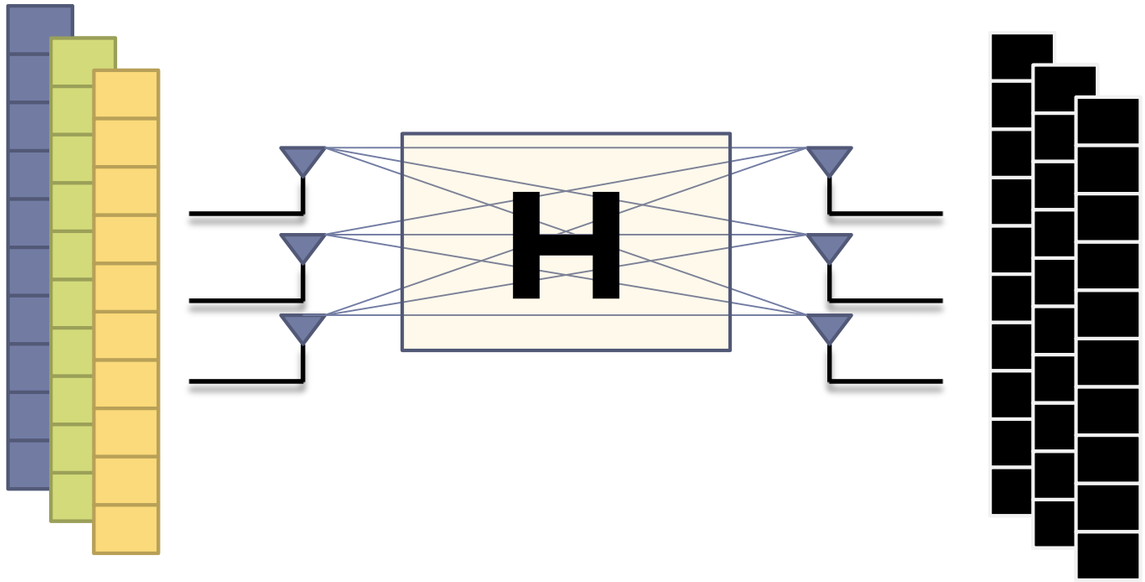


$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} n_0 \\ n_1 \\ n_2 \end{bmatrix}$$

$$\mathbf{y} = \mathbf{Hx} + \mathbf{n}$$

- Spatial Multiplexing: ↑ throughput by transmitting multiple streams
- Receiver: Transmit streams interfere with each other

MIMO-OFDM



Break a wideband signal into many independent subcarriers

Perform MIMO detection independently many times, one per subcarrier

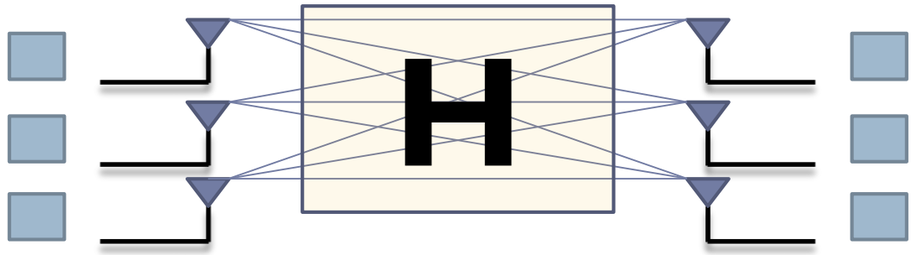
Many subcarriers for many wireless standards.

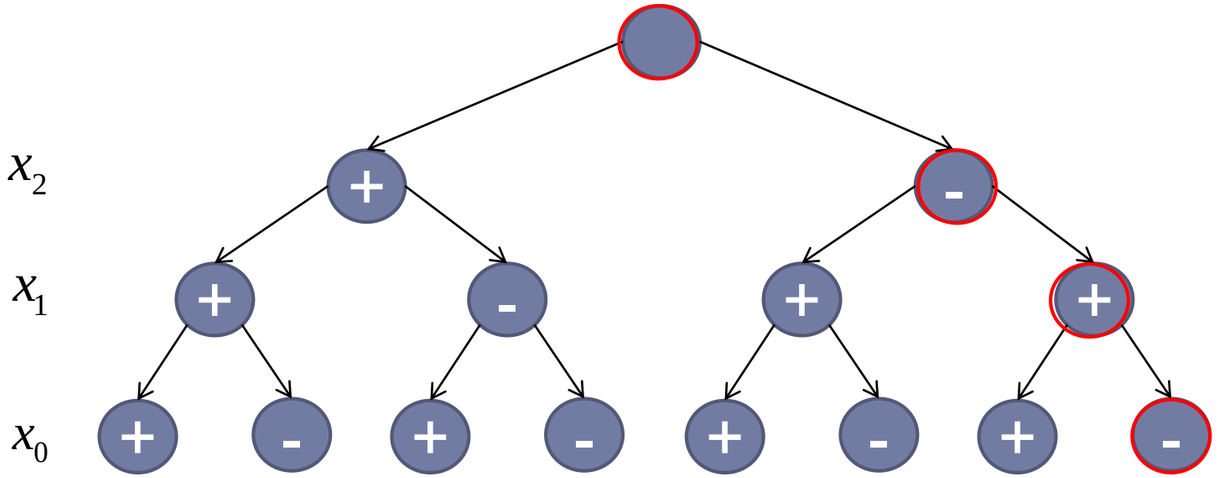
- LTE 20Mhz subframe: 14*1200 subcarriers

MIMO Detection

$$\begin{bmatrix} \hat{y}_{00} \\ \hat{y}_{11} \\ \hat{y}_{22} \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_{00} \\ x_{11} \\ x_{22} \end{bmatrix} + \begin{bmatrix} n_{00} \\ n_{11} \\ n_{22} \end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{Hx} + \mathbf{n}$$





Search Space for 3x3 BPSK

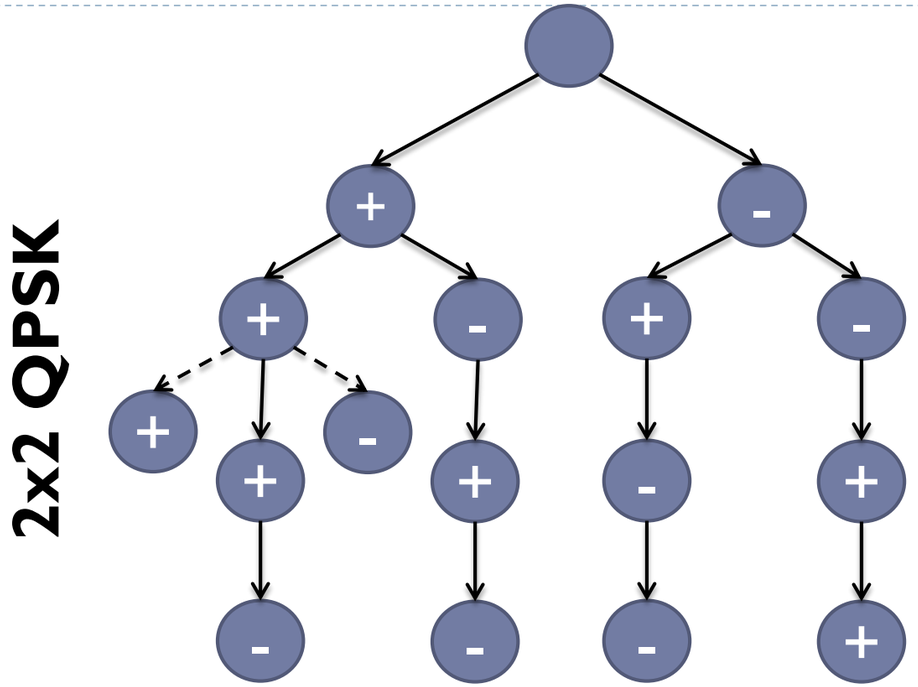
$$\boldsymbol{d_2} = \|\hat{\boldsymbol{y}}_2 - \boldsymbol{r_{22}}(x_2)\|^2$$

$$\boldsymbol{d_1} = \|\hat{\boldsymbol{y}}_1 - \boldsymbol{r_{12}}(x_2) - \boldsymbol{r_{11}}(x_1)\|^2$$

$$\boldsymbol{d_0} = \|\hat{\boldsymbol{y}}_0 - \boldsymbol{r_{02}}(x_2) - \boldsymbol{r_{01}}(x_1) - \boldsymbol{r_{00}}(x_0)\|^2$$

- Probability of a path, x, is **inversely prop.** to $\mathrm{d}(\hat{\mathrm{y}}, x) = \|\hat{\mathrm{y}} - Rx\| = \sum_i \boldsymbol{d_i}$
- Probabilities of all paths are used to generate bit probability values.
  - 4x4 64QAM→64^4 = 16,777,216 paths

SSFE Detector



1st antenna Real

1st antenna Imag

1st level: enumerate all modulation points.

2nd antenna Real

2nd antenna Imag

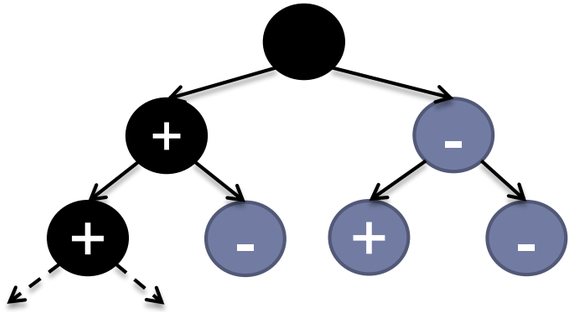
Subsequent levels: depth-first search, pick the best outgoing node

Selective Spanning with Fast Enumeration (SSFE)*

- Real value decomposition
- Data parallel deterministic search

Generate M likely paths which are used to generate bit probability values

SSFE Detector: Node Expansion



Inputs:

- **P**: [x_2=1,x_1=1]
- Channel gains: **r** : [r_{00} r_{01} r_{02}]
- Received signal: y_0

- Find best node–x_0 that minimizes the cumulative distance
  - Pick the constellation point closest to the zero forcing solution.

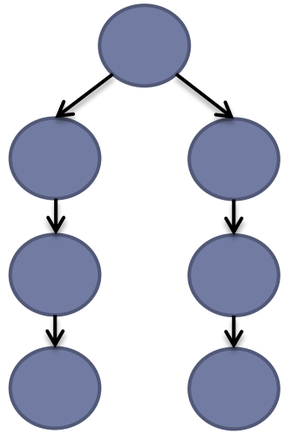
Zero forcing solution: $\hat{x}_0 = \frac{1}{r_{00}}(\hat{y}_0 - r_{02}x_2 - r_{01}x_1)$

64 QAM example: $\hat{x}_0 = 3.9$

Schnoor-Euchner enumeration

GPU Implementation

- Search algorithm maps well onto GPU
  - Data parallel with no sharing
    - Each search path is independent
  - Efficient node expansion
    - complexity doesn't depend on modulation
  - Modest storage requirement



M threads per detection

GPU Implementation of SSFE

```
//enumerate a modulation point for 1st antenna
path[0] = mod_table[tid%8];
path[1] = mod_table[tid/8];
dist+= calc_dist(y, r, path[0]);
dist+= calc_dist(y, r, path[1]);

//depth first search
For i = 2:ntx
    //compute partial Euclidean dist
    ped = 0;
    For j = 0:i
        ped += calc_dist(y, r, path[j]);
    //find best outgoing path
    Path[j] = SE_expand(dist, r);
    dist = update_dist(dist, ped);
```

One thread block handles one subcarrier

- Spawns 1 thread per modulation point (M threads)
- Completely unrolled inner and outer loops
- Path stored in registers

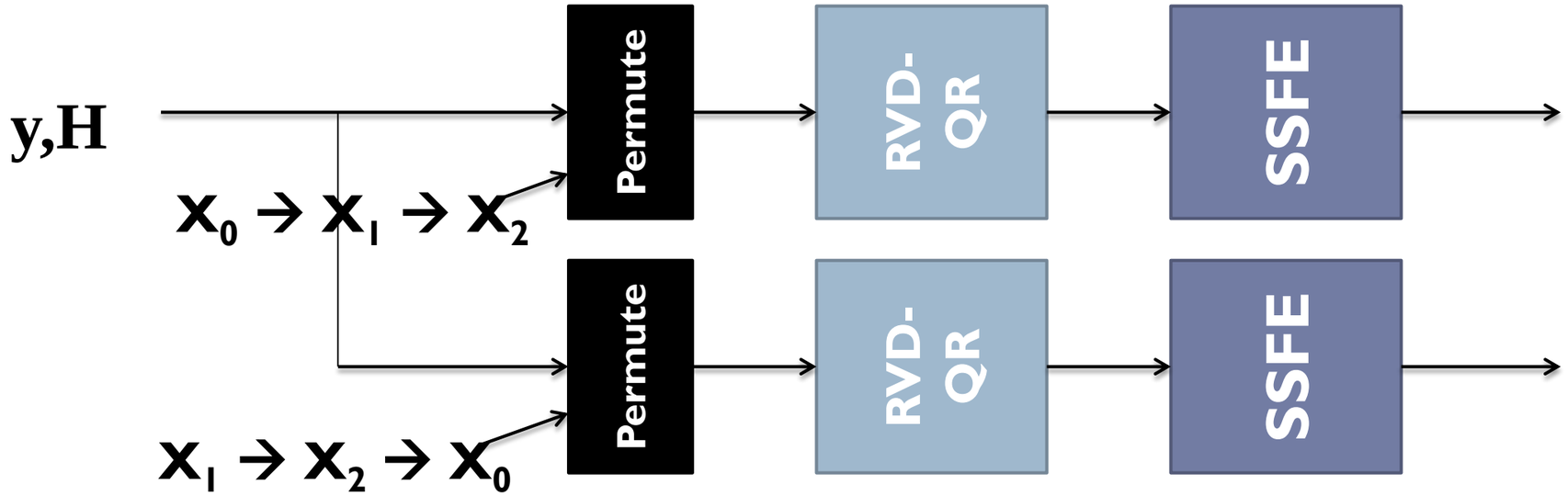
Demodulator + soft estimate computation not shown

Result

- 4x4 16QAM: 940Mbps
- 4x4 64QAM: 480Mbps

N-Way MIMO Detector

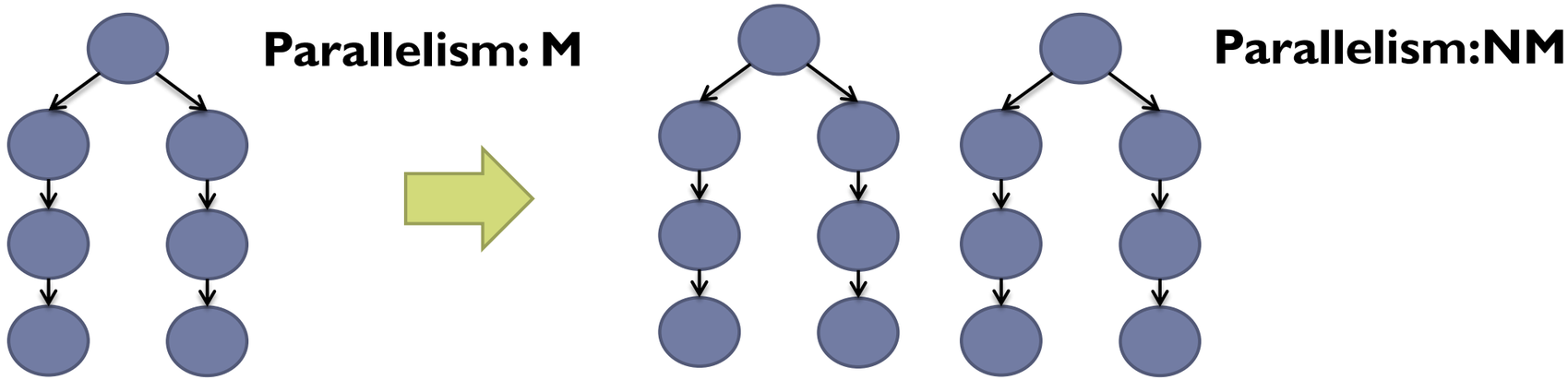
- Duplicate search block depending on FER requirement.
  - Add permute block which enforces a detection order
  - Example: N = 2, two search blocks
  - Larger lists, NM candidates



Qi, Q., Chakrabarti, C, *Parallel high throughput soft-output sphere decoder*, (SIPS 10)
M Wu, C Dick, JR Cavallaro, *Improving MIMO sphere detection through antenna detection order scheduling* (SDR Forum 11)

GPU Implementation of *N*-Way MIMO Detector

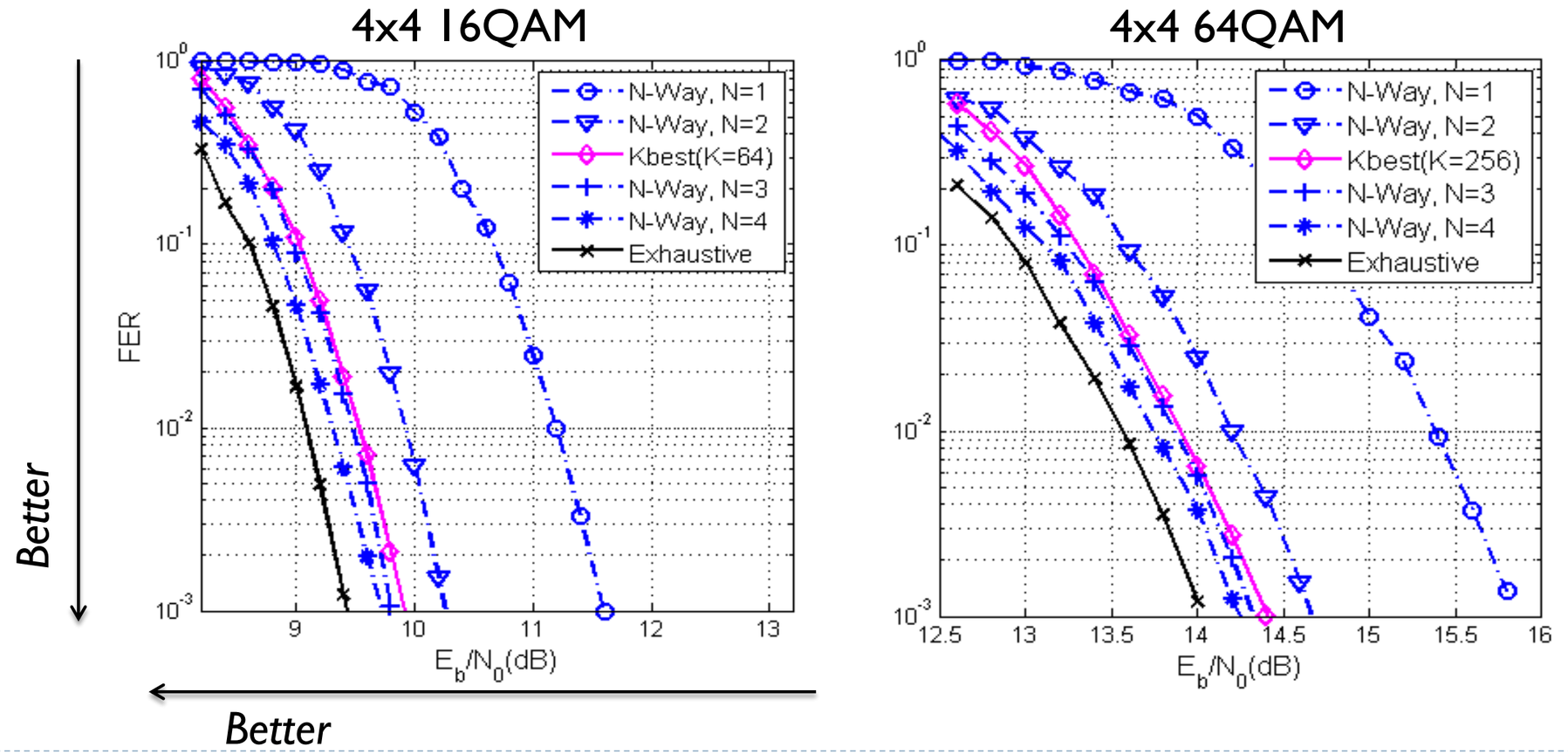
- Duplicate threads to improve accuracy of the detection algorithm



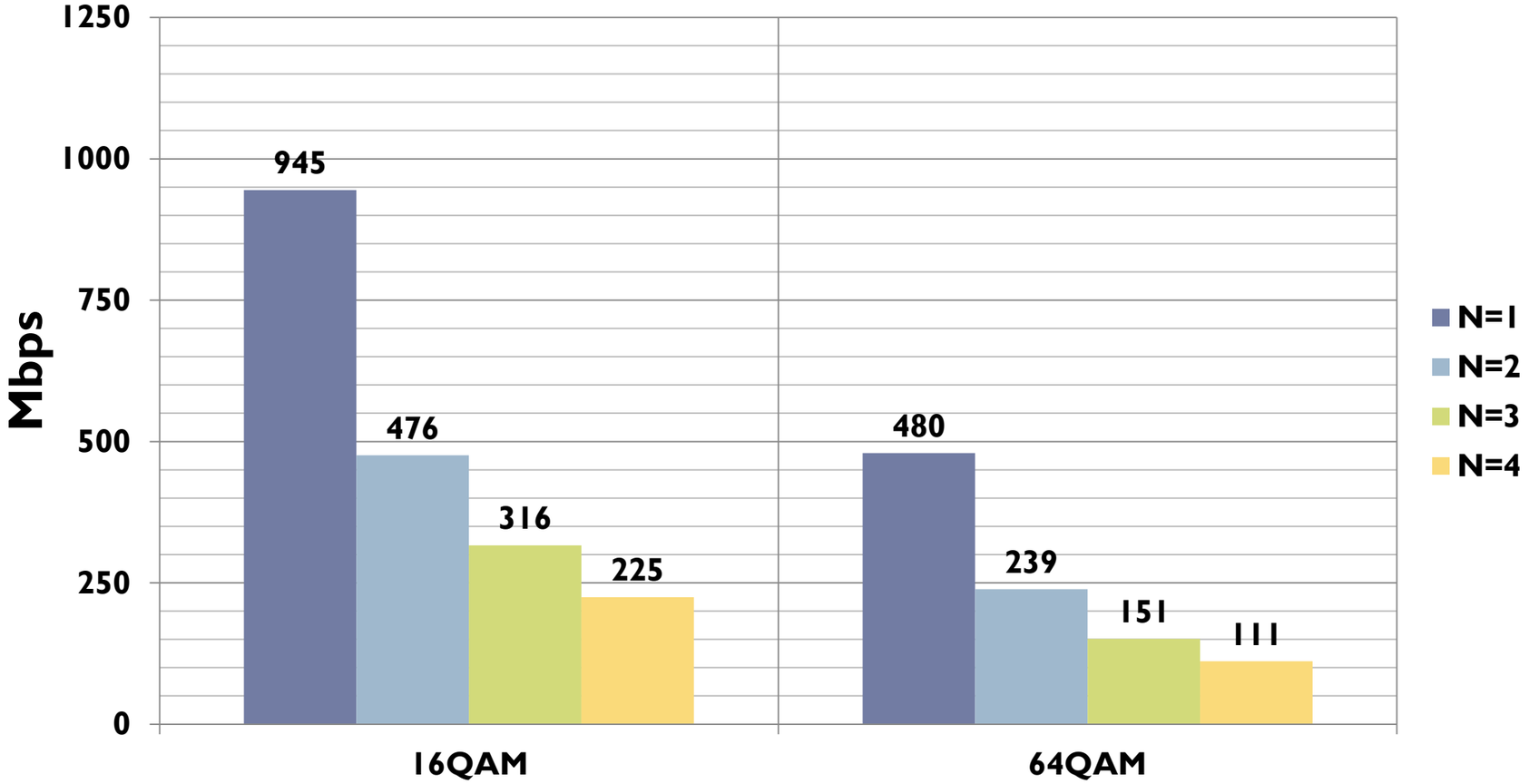
- Divide a thread block into N subsections
- Each subsection consists of M threads operates on a different channel permutation
  - Performs SSFE detection independently
  - Generate a NM size candidate list.

N-Way MIMO FER Performance

- Soft Output detectors + Rate 1/2 WiMAX LDPC code, Rayleigh fading channel.
  - 1 outer iteration + 20 inner iteration with early termination
  - Compared to soft-output K-best and exhaustive (MAP) detector.



N-Way MIMO Detector Throughput



- GK104, 1536 SM @ 915MHz, 256-bit DDR5 @ 6Gbps
- 8192 subcarriers, Kernel time only

Performance Comparison

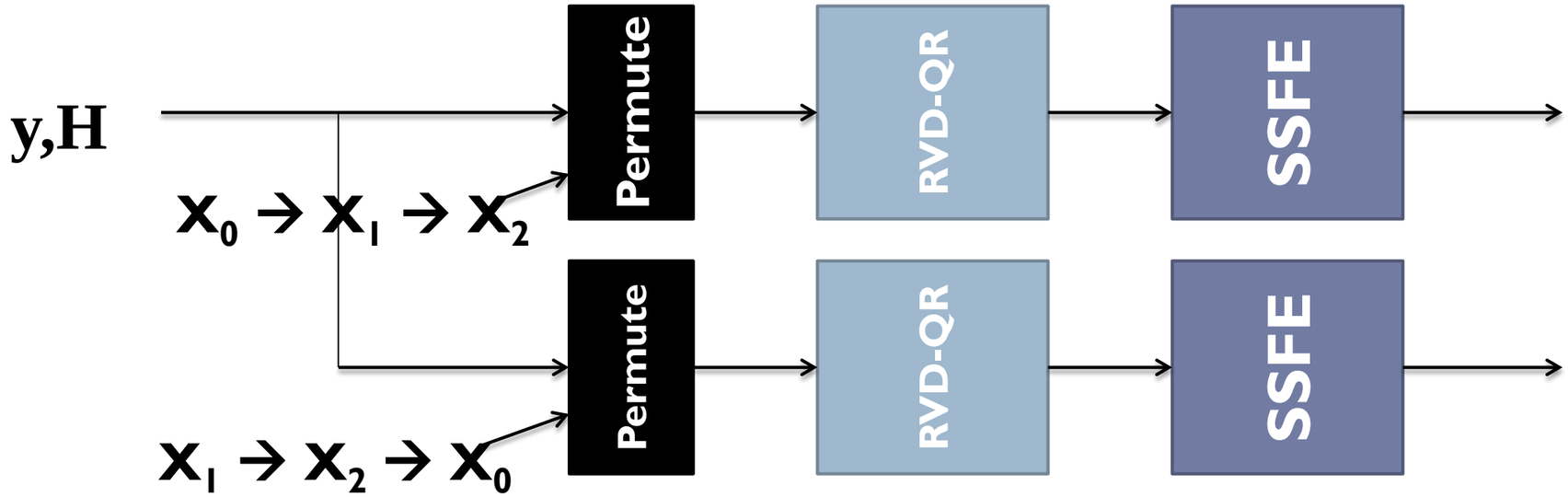
Number of Subcarriers	16QAM (Mbps)		64QAM (Mbps)	
	FPFSD*	Ours N=4	FPFSD*	Ours N=4
150*7	72.41	232.27	18.6	67.55
300*7	82.56	246.60	18.59	69.12
600*7	90.44	256.83	17.9	69.53
900*7	91.39	256.22	17.4	70.05
1200*7	92.31	256.88	17.2	69.97

- FPFSD
  - N=4—4 parallel detectors with different permutations
  - Differences: a) operates in complex domain b) one kernel for search + one kernel for soft-output generation
- Fermi, 448 core @ 1150MHz, 320bit DDR5 @ 3Gbps

*Sandra Roger, et.al *Fully Parallel GPU Implementation of a Fixed-Complexity Soft-Output MIMO Detector* (IEEE TVT 2012)

N-Way MIMO Detector

- Duplicate search block depending on FER requirement.
  - Add permute block which enforces a detection order
  - Example: N = 2, two search blocks
  - Larger lists, NM candidates



Qi, Q., Chakrabarti, C, *Parallel high throughput soft-output sphere decoder*, (SIPS 10)
M Wu, C Dick, JR Cavallaro, *Improving MIMO sphere detection through antenna detection order scheduling* (SDR Forum 11)

N-Way QR Decomposition

Divide a thread block into N subsections

Each subsection of N threads operates on a different channel permutation

- Performs modified Gram Schmidt QR on an extended matrix $[H|Y]$
- Generate R and $\hat{y}$

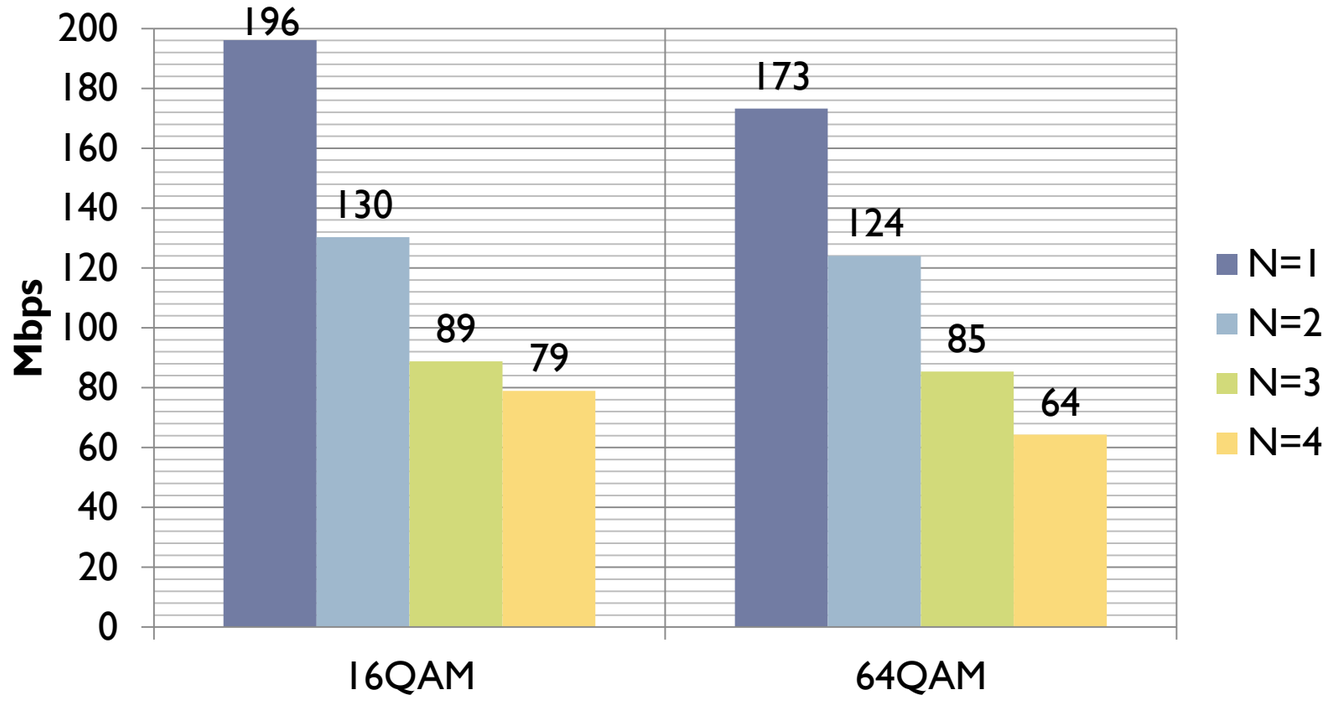
QR decomposition time for 8192 symbols

	N=1	N=2	N=3	N=4
4x4	0.201 ms	0.350 ms	0.551ms	0.675 ms

- GK104, 1536 SM @ 915MHz, 256-bit DDR5 @ 6Gbps
- Kernel time only

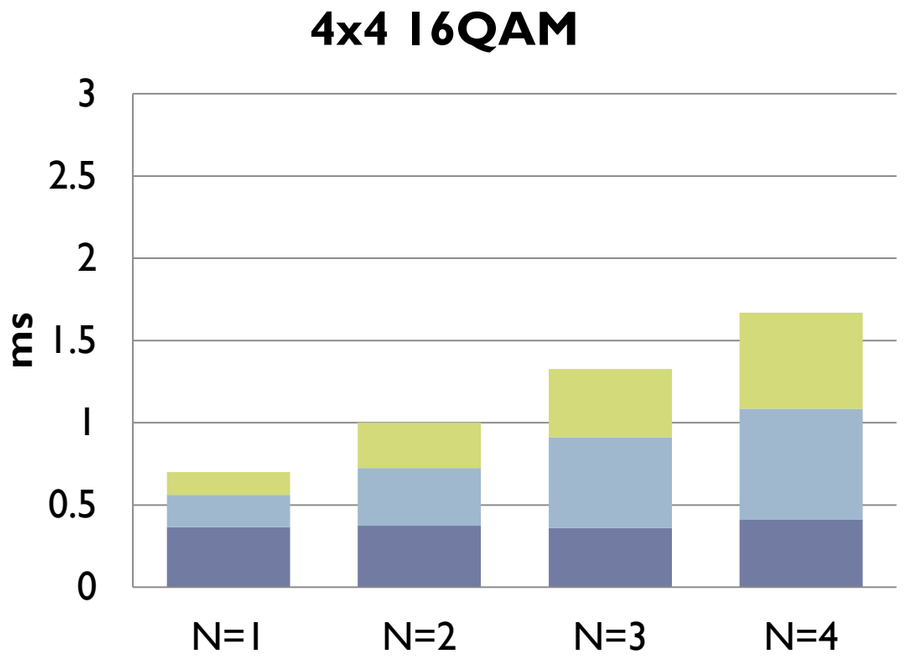
Complete Design

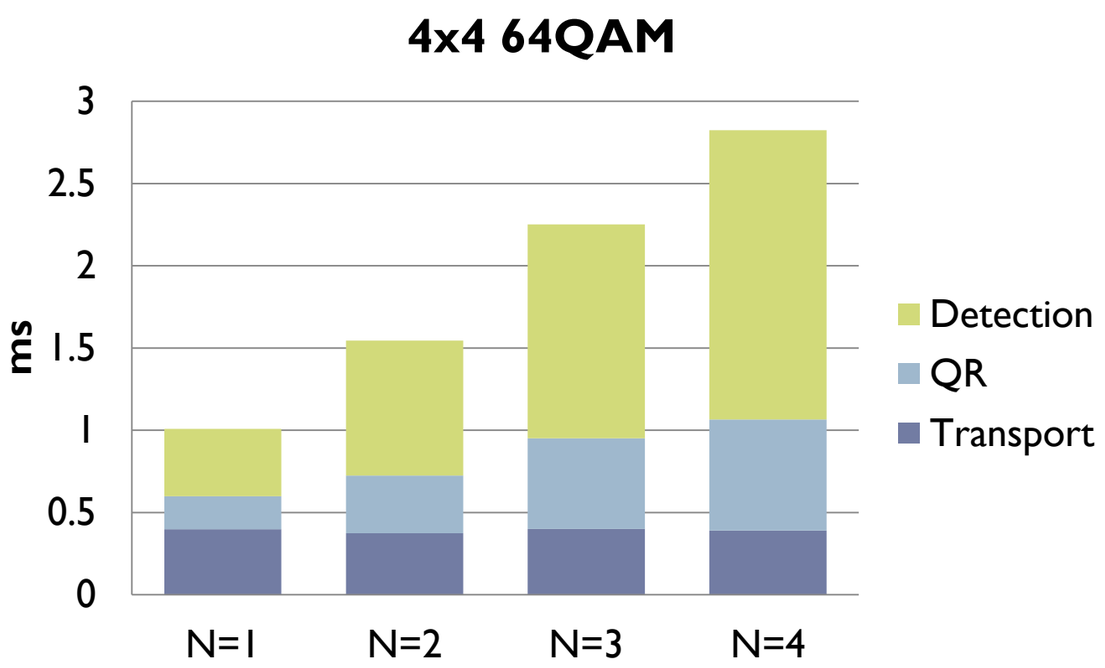
- Complete design, QR + MIMO Detection
- Also includes PCIE transfer time



- GK104, 1536 SM @ 915MHz, 256-bit DDR5 @ 6Gbps
- 8192 subcarriers

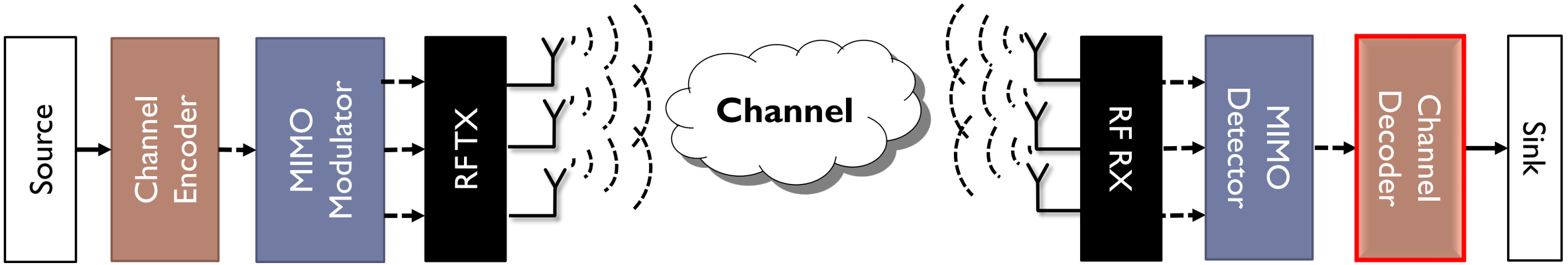
Complete Design





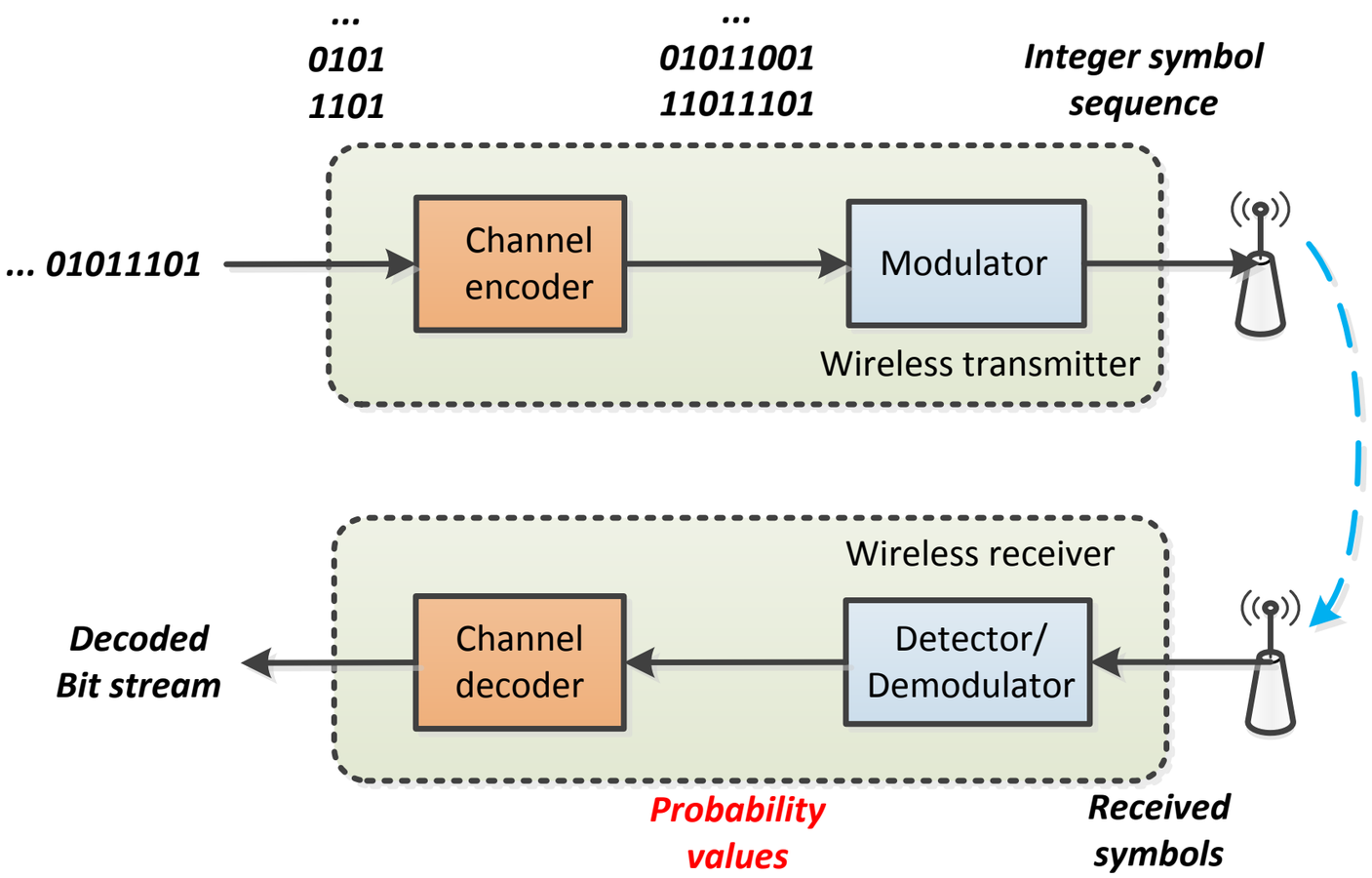
- Transfer time doesn't depend on N (# of parallel search) or M (modulation size)
  - Transfer time can be hidden
- QR depends only on N
- Detection depends on N and M

Outline



- Multi-standard LDPC Decoder on GPU
  - Introduction to LDPC decoding algorithm
  - Kernel mapping
  - Optimization techniques
  - Experiment results

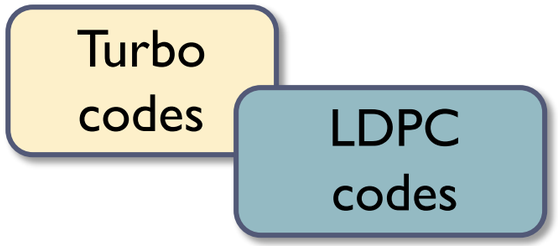
Channel Coding



- Linear block codes
- Encoding: $\boldsymbol{x} \cdot \boldsymbol{G} = \boldsymbol{c}$
  - K-bit $\boldsymbol{x}$ encoded into N-bit codeword $\boldsymbol{c}$ (K<N)
  - Generator matrix $\boldsymbol{G}$
  - Parity check matrix $\boldsymbol{H}$: $\boldsymbol{H} \cdot \boldsymbol{cT} = \boldsymbol{0}$ $(\boldsymbol{G} \cdot \boldsymbol{H^T} = \boldsymbol{0})$

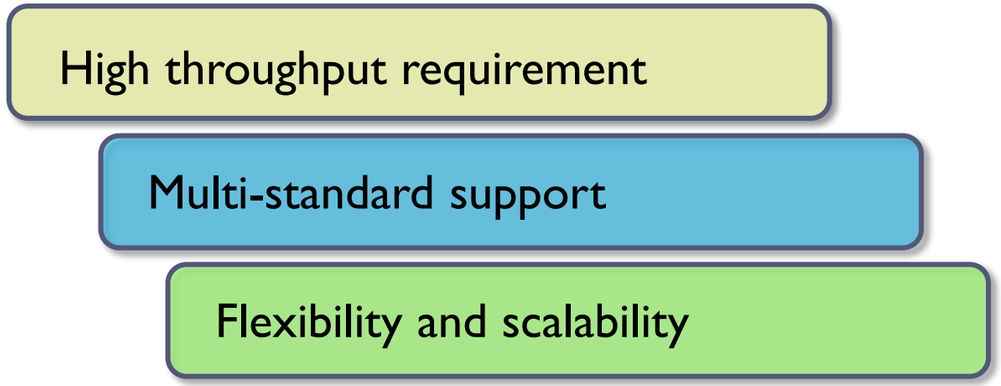
Low-density parity-check (LDPC) codes

- Error-correction codes
- Provides near-capacity error-correcting performance



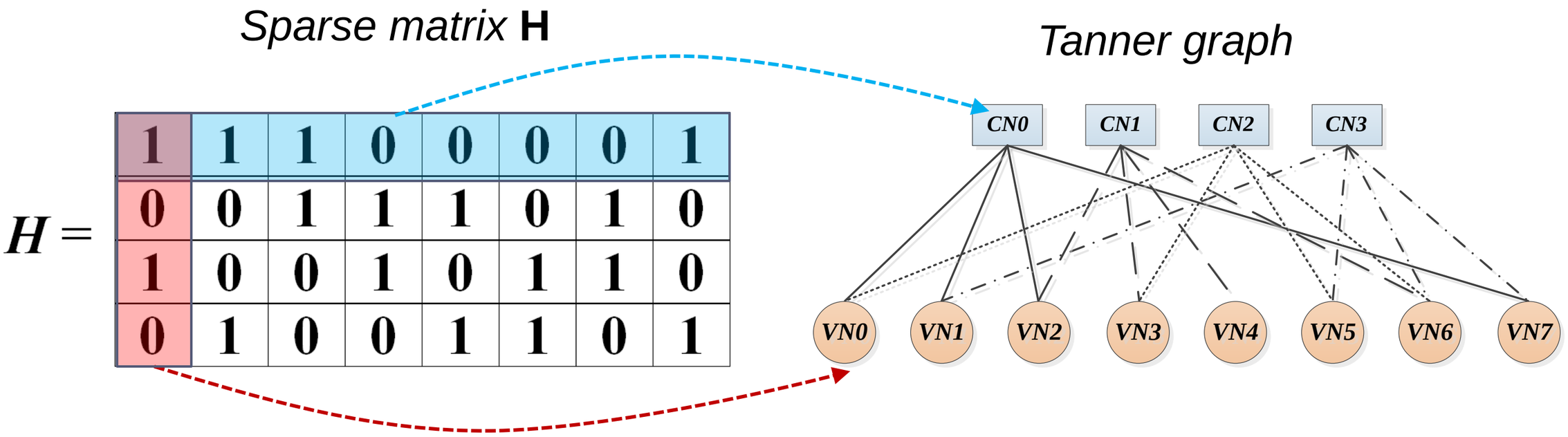
- Application of LDPC codes
  - Wireless communication
    - IEEE 802.16m WiMax
    - IEEE 802.11n, 802.11ac WiFi
  - 10Gbps Ethernet communication
    - IEEE 802.3an
  - Digital broadcast: DVB-S2
  - High speed magnetic storage device
  - Satellite communication

Challenges of decoder design

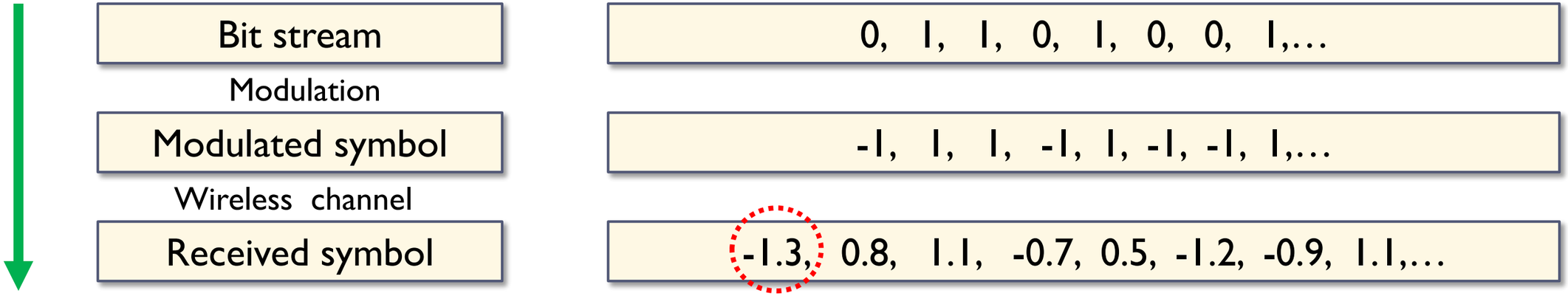


LDPC Codes

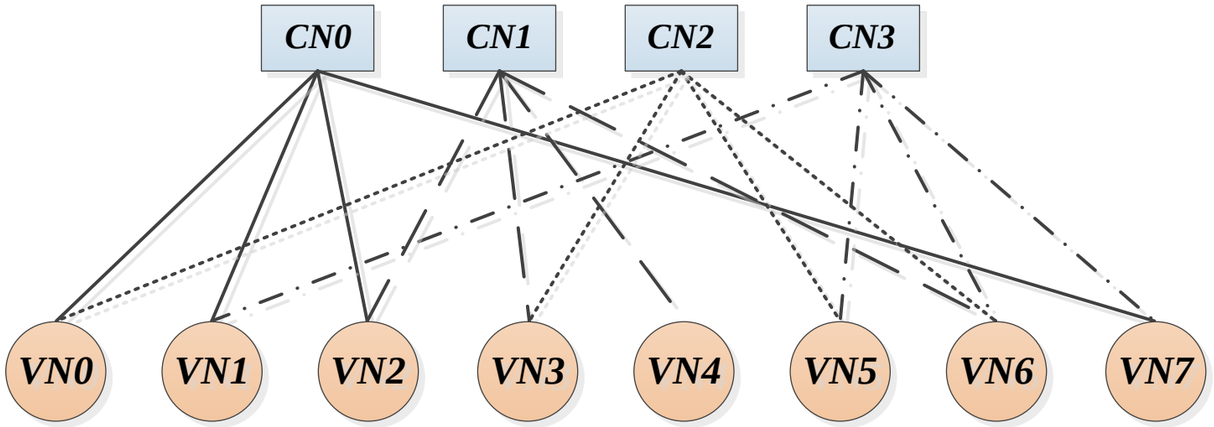
- LDPC codes are linear block codes defined by **sparse matrices** $\boldsymbol{H}$
- Codeword $\boldsymbol{c}$ should satisfy the parity-check equations: $\boldsymbol{H} \cdot \boldsymbol{c}^T = 0$
- Belief propagation decoding algorithm



LDPC Decoding: Belief propagation decoding

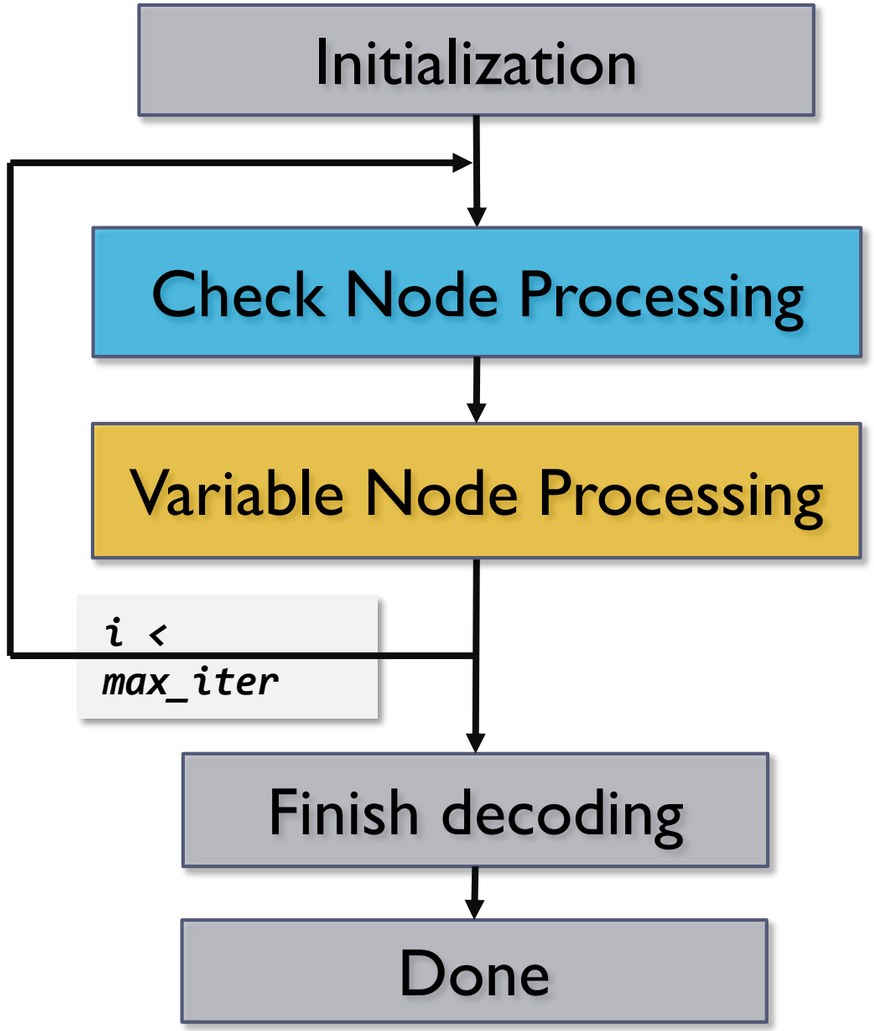


Probability(c_n=0 | received) **VS** *Probability(c_n=1 | received)*



- $\boldsymbol{H} \cdot \boldsymbol{c}^T = 0$
- Complexity ~O(N^3)

Belief propagation decoding



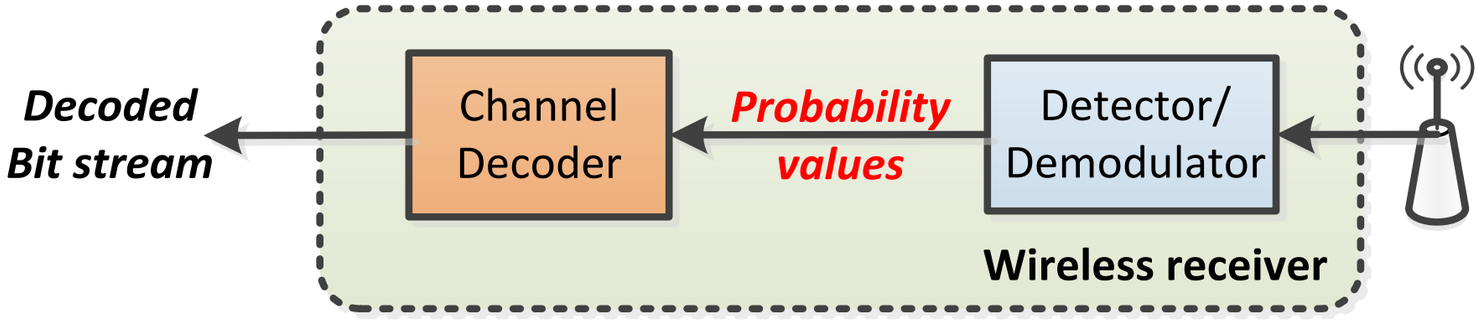
Check Node Processing, *Rmn*

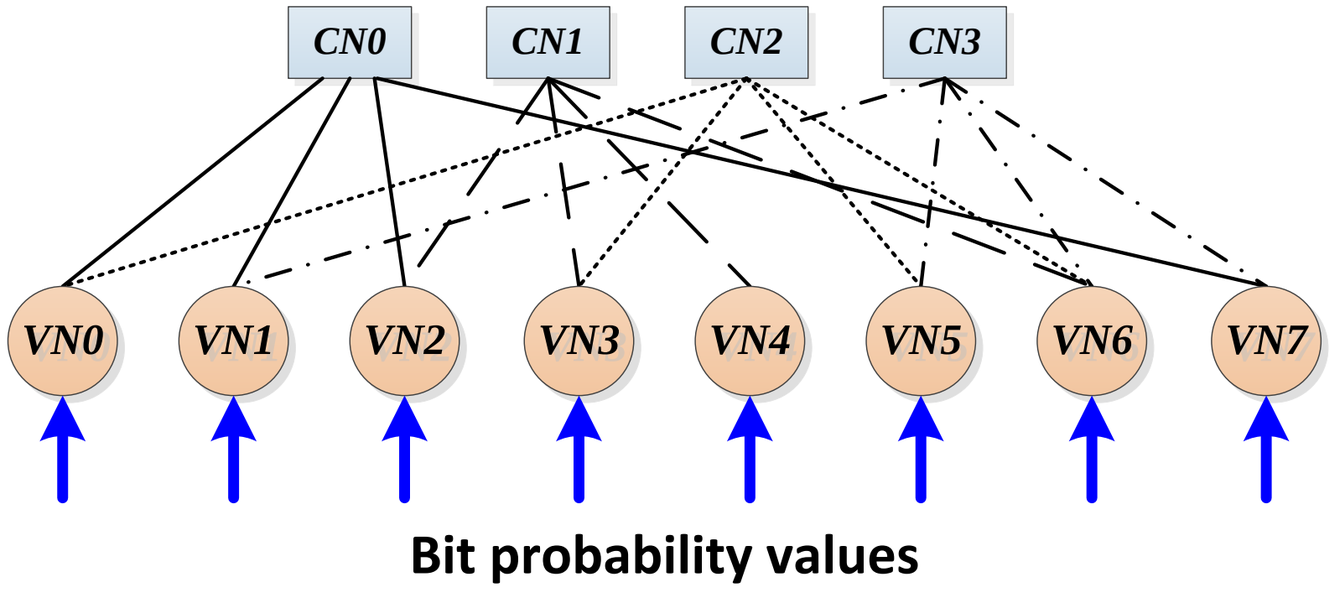
$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Variable Node Processing, *Qmn*

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Belief propagation decoding algorithm: initialization

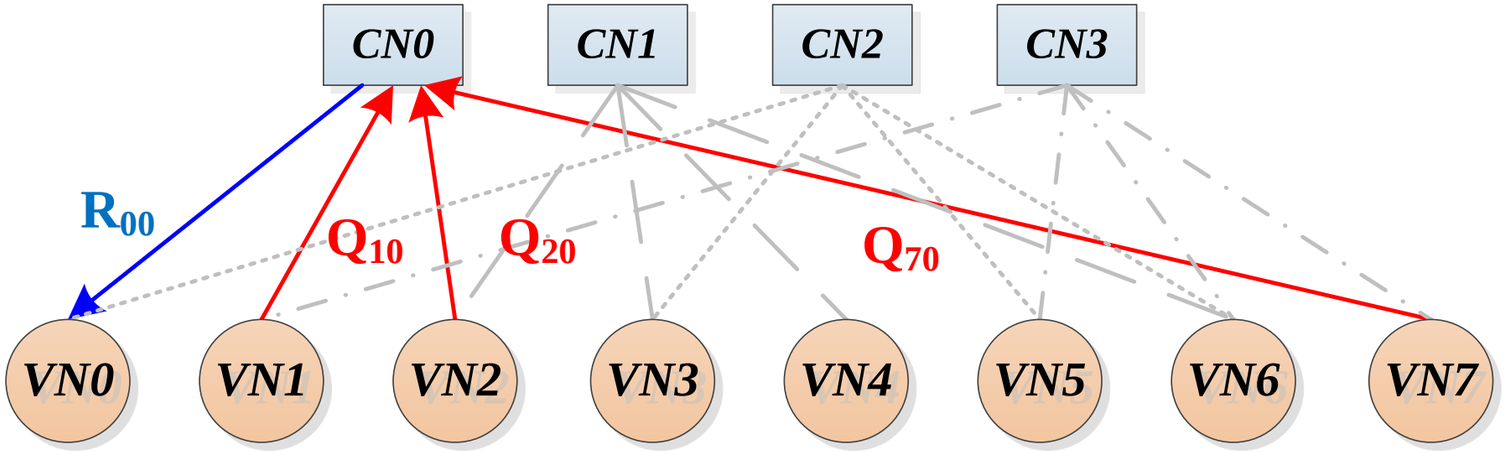


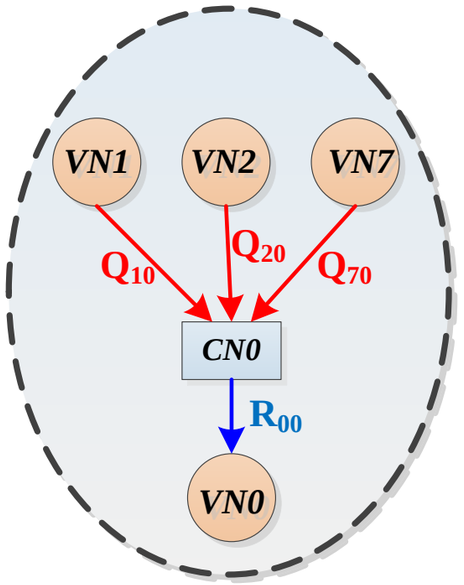


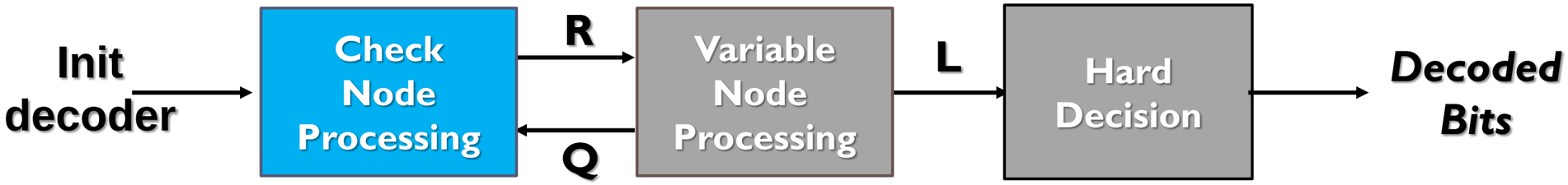
Belief propagation decoding algorithm: CNP

$H =$

1	1	1	0	0	0	0	1
0	0	1	1	1	0	1	0
1	0	0	1	0	1	1	0
0	1	0	0	1	1	0	1



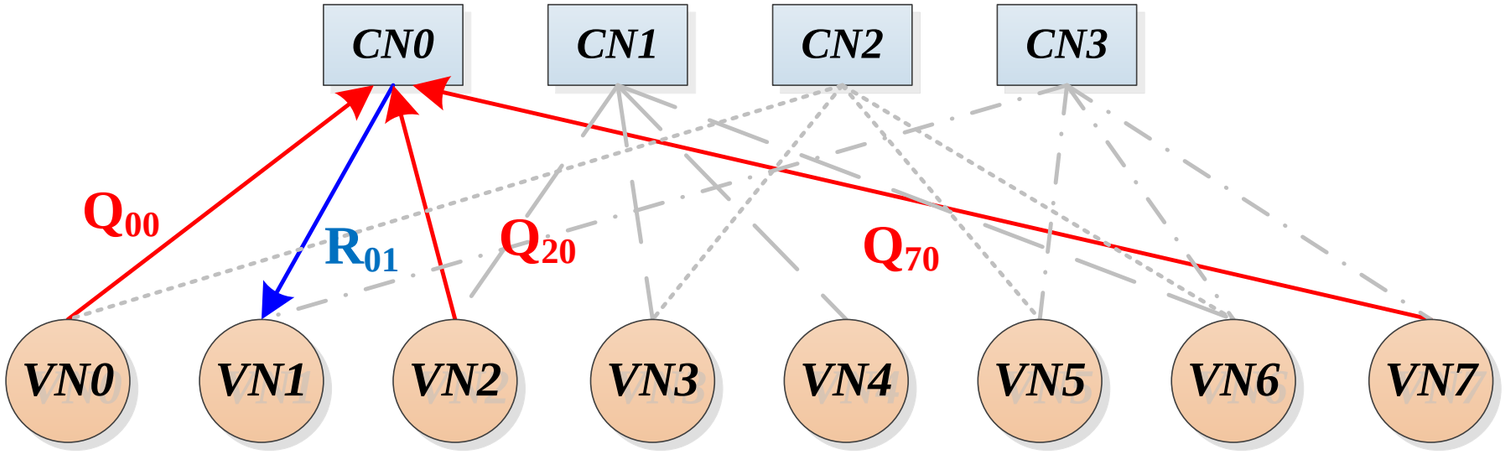


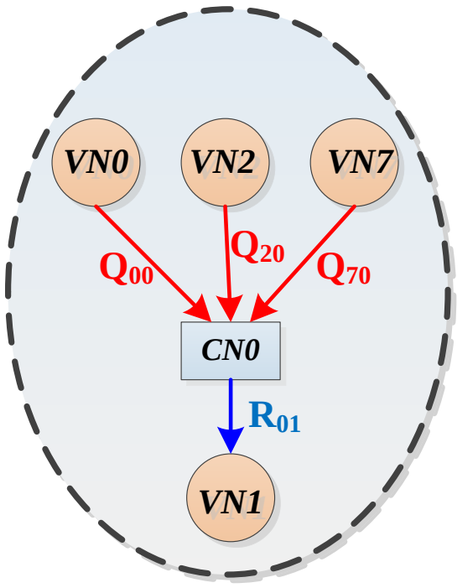


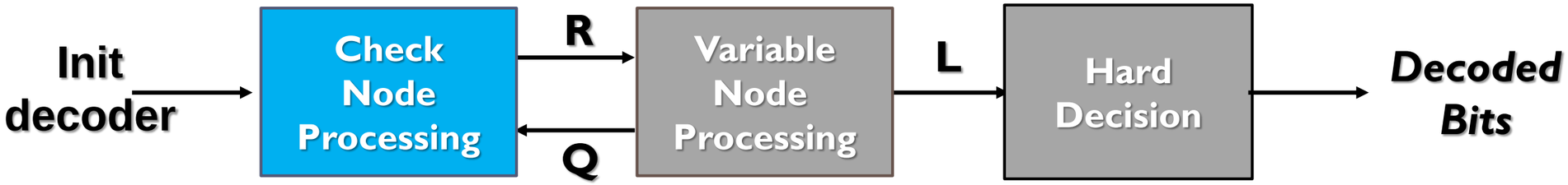
Belief propagation decoding algorithm: CNP

$H =$

1	1	1	0	0	0	0	1
0	0	1	1	1	0	1	0
1	0	0	1	0	1	1	0
0	1	0	0	1	1	0	1



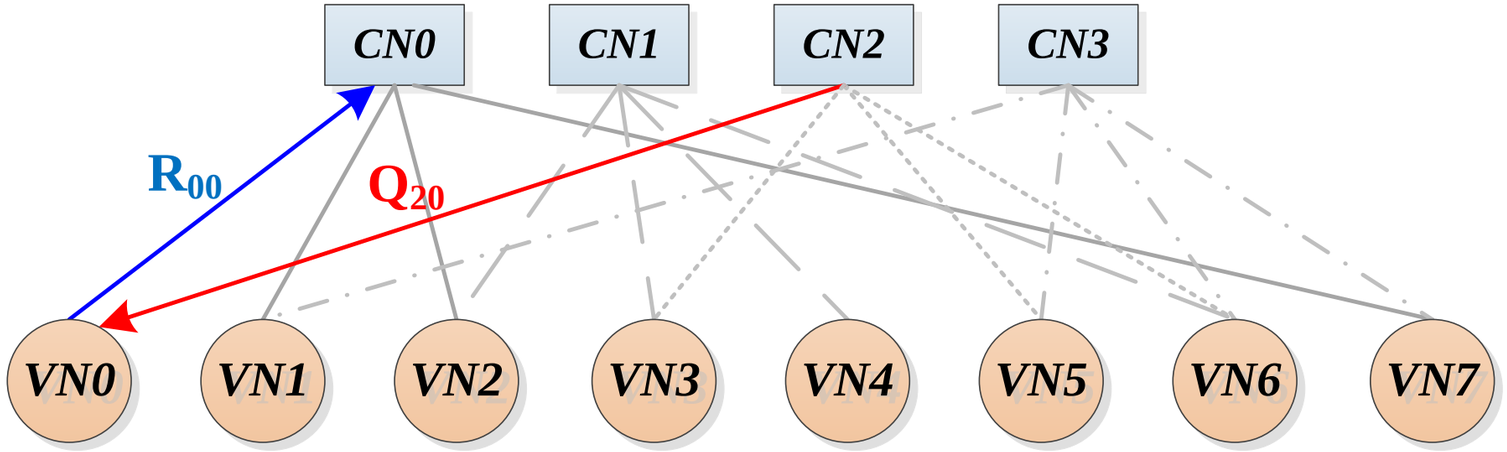


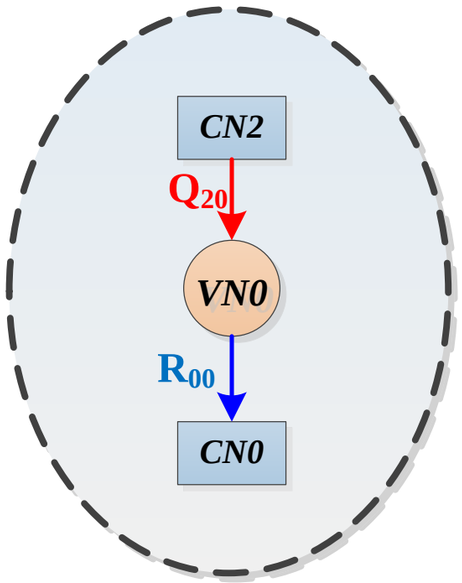


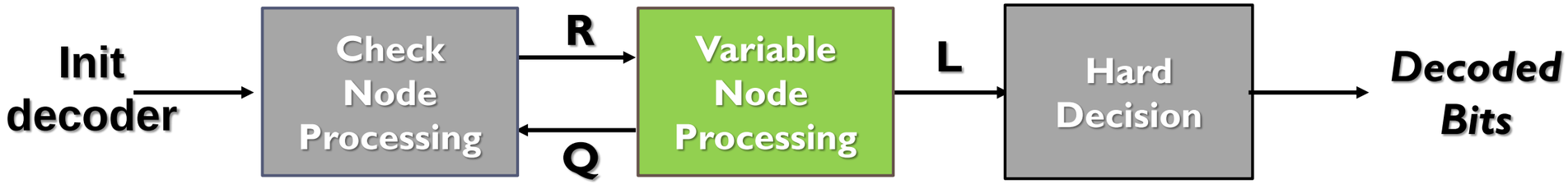
Belief propagation decoding algorithm: VNP

$H =$

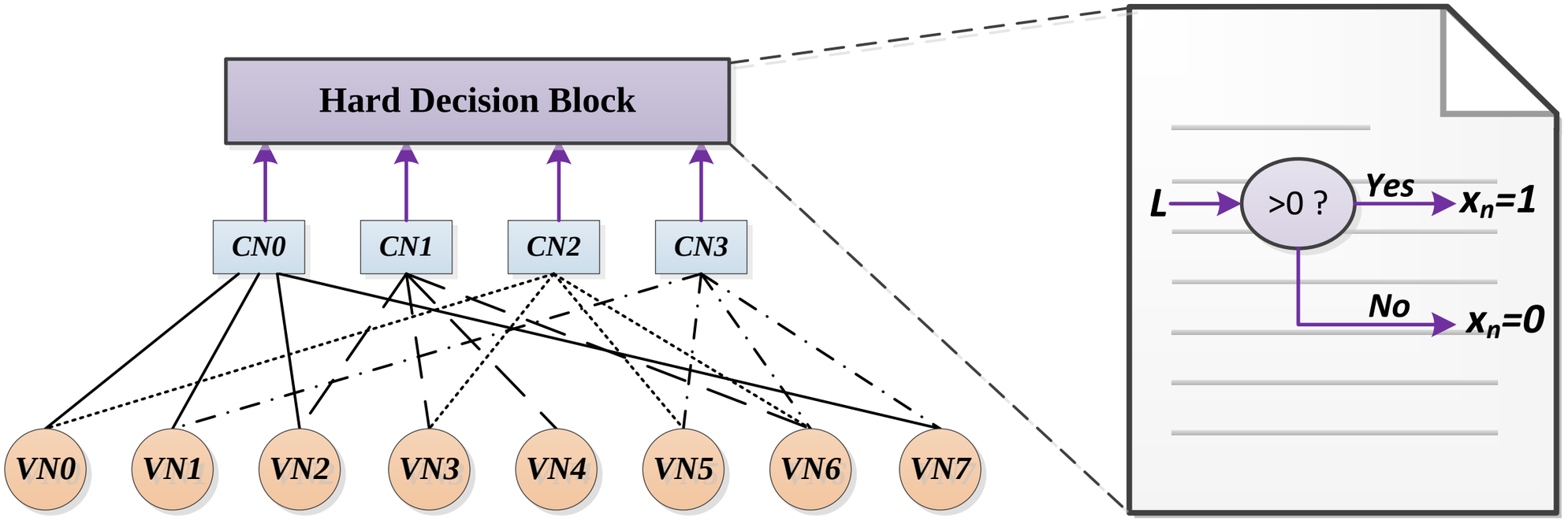
1	1	1	0	0	0	0	1
0	0	1	1	1	0	1	0
1	0	0	1	0	1	1	0
0	1	0	0	1	1	0	1

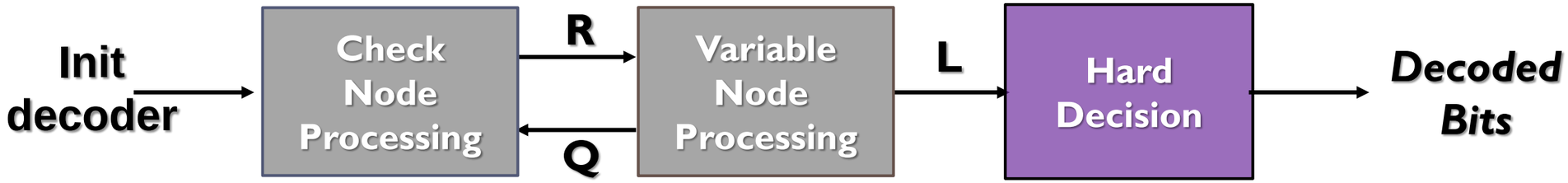






Belief propagation decoding algorithm: hard decision

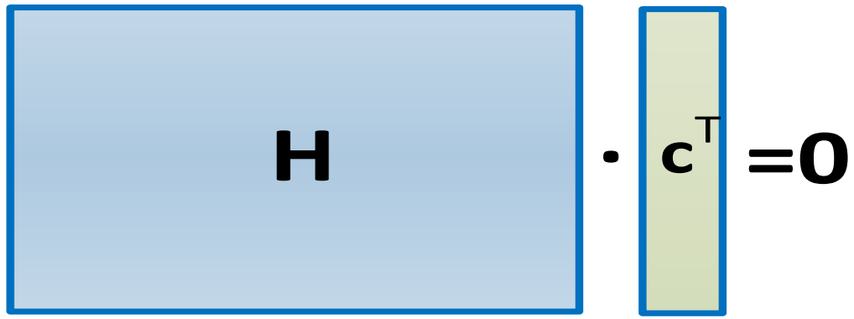




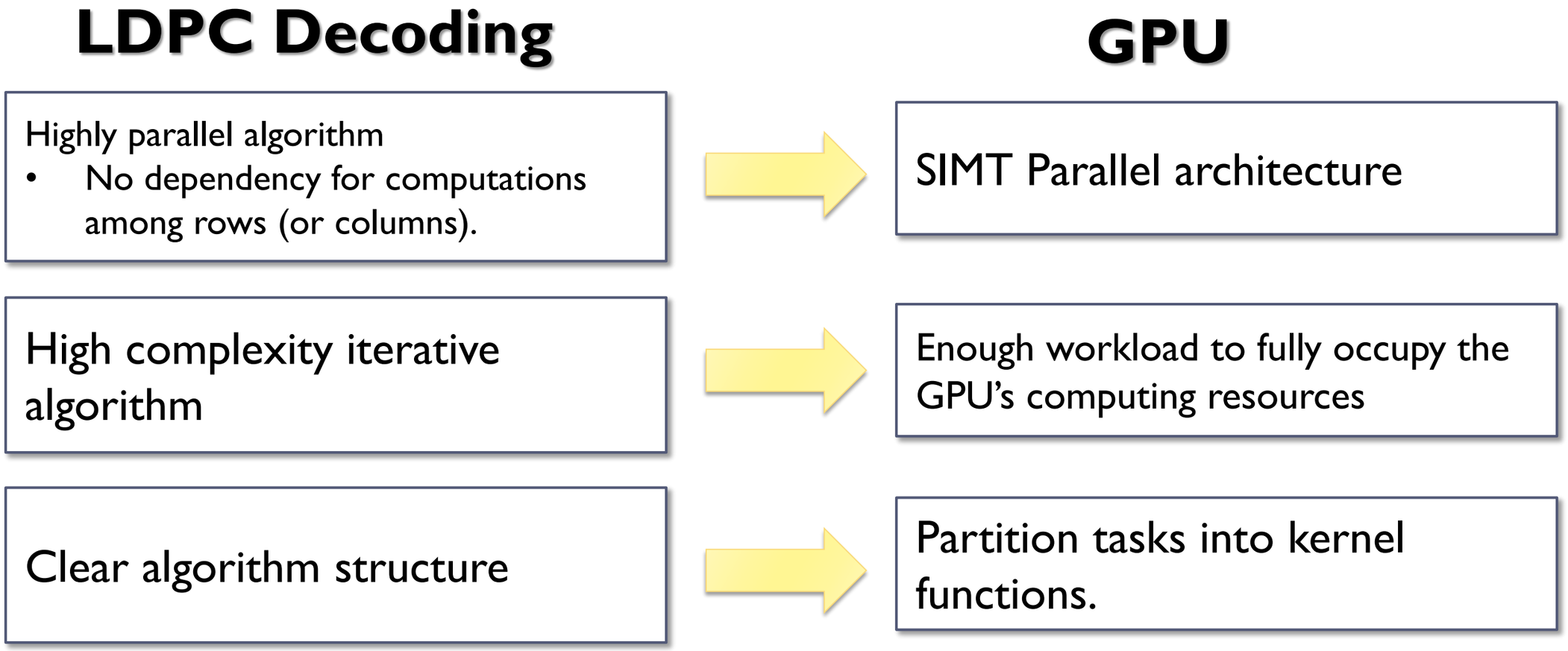
Early Termination

- Early termination (ET)
  - Avoid unnecessary computations when codeword converges
  - Widely used in low power decoding architecture.

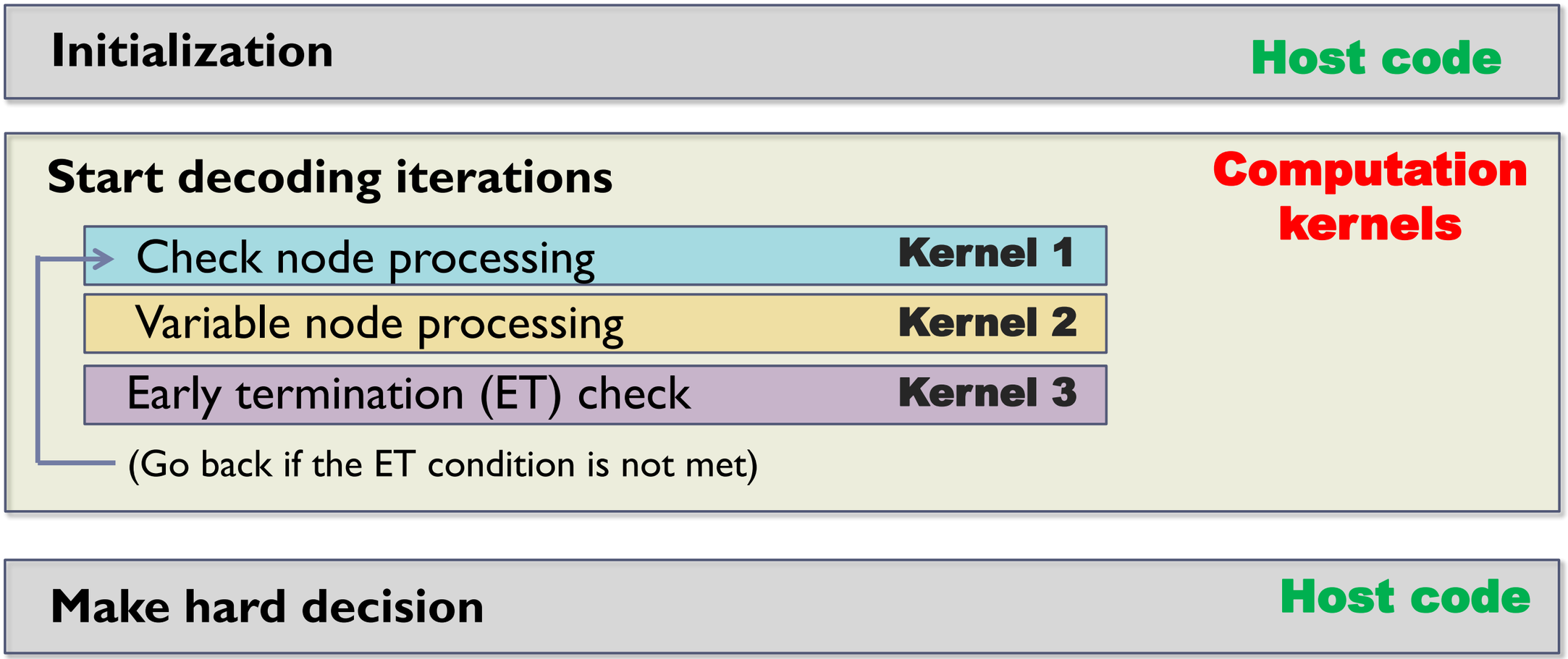
- ET for LDPC decoder
  - Use **parity check equation:** $\boldsymbol{H\ c^T=0}$
  - Use **massive threads** to perform parity check



Why GPU for LDPC Decoding?

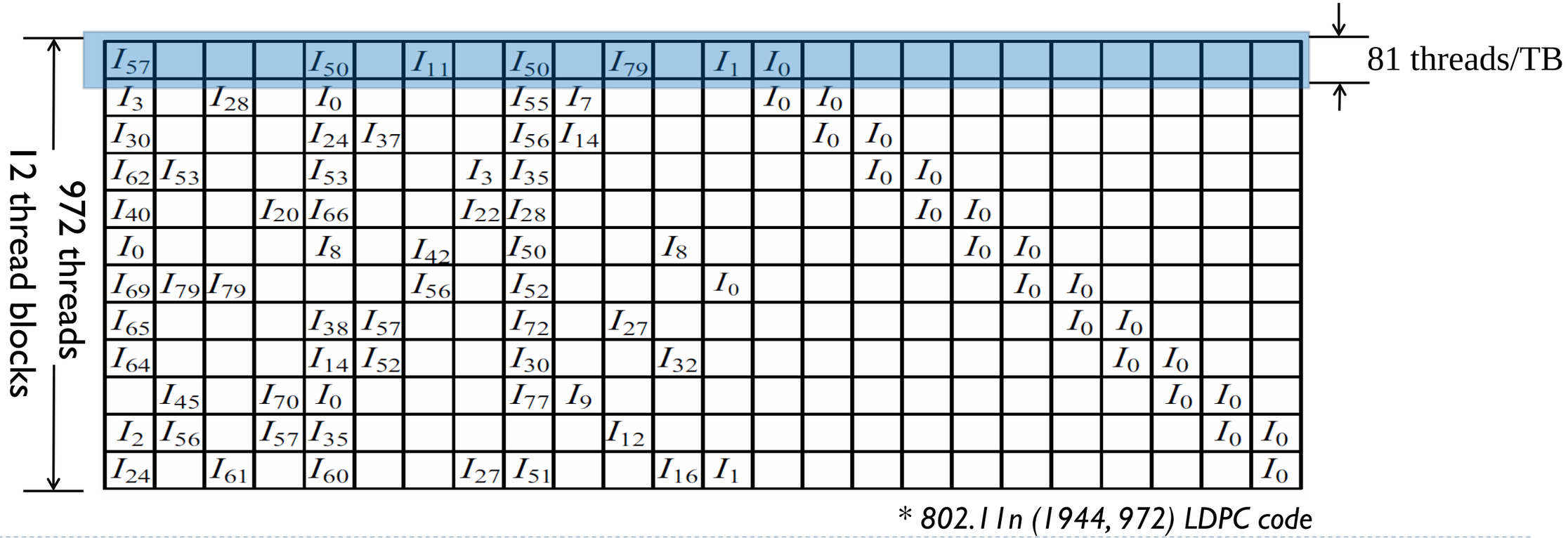


Partition the LDPC Decoding Task



CUDA Kernel 1: Check Node Processing

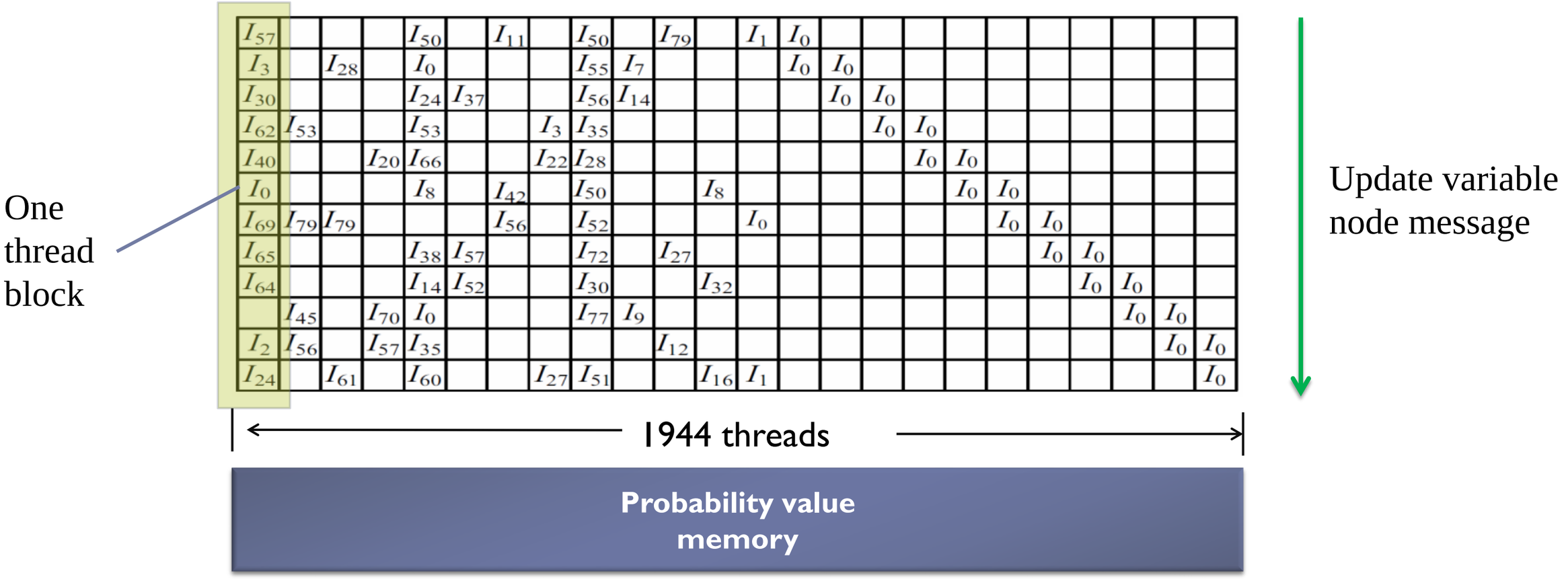
- One thread block processes one row of sub-matrices
- Each thread block contains 81 threads, each thread processes one row of the H matrix (**one check node**).



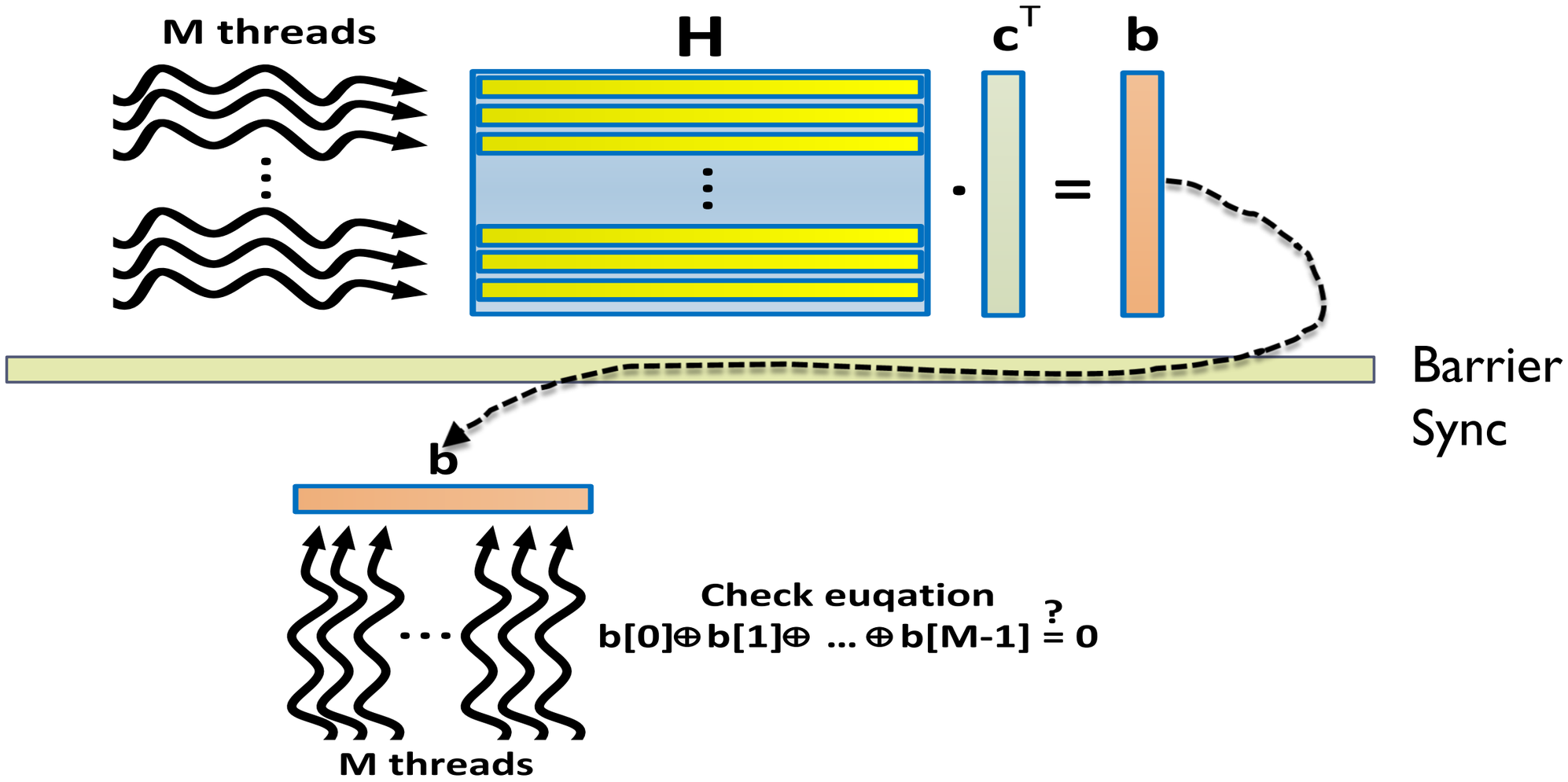
** 802.11n (1944, 972) LDPC code*

CUDA Kernel 2: Variable Node Processing

- One thread block processes one column of sub-matrices.
- Use 1944 threads to run concurrently.

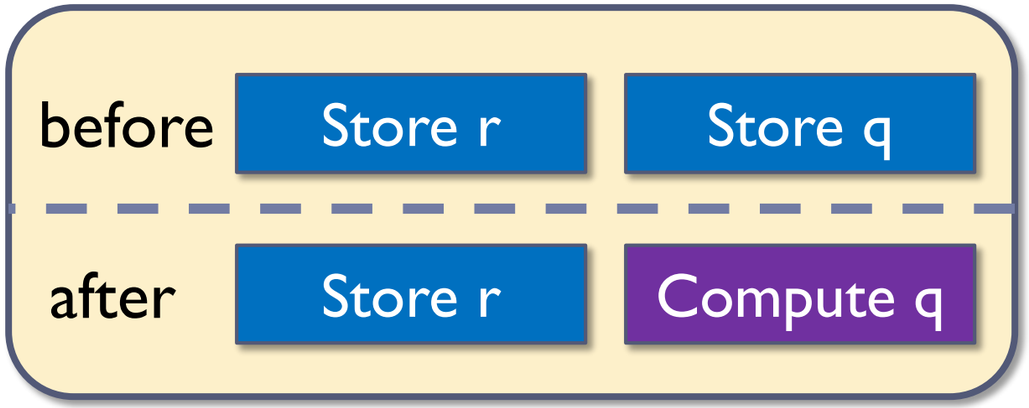


CUDA Kernel 3: Parallel Early Termination

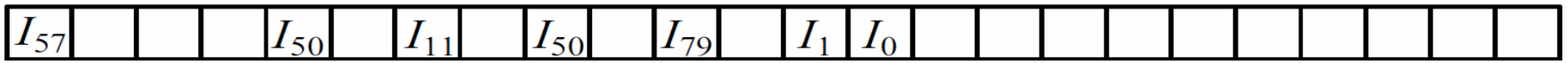


Decoding Algorithm Optimization

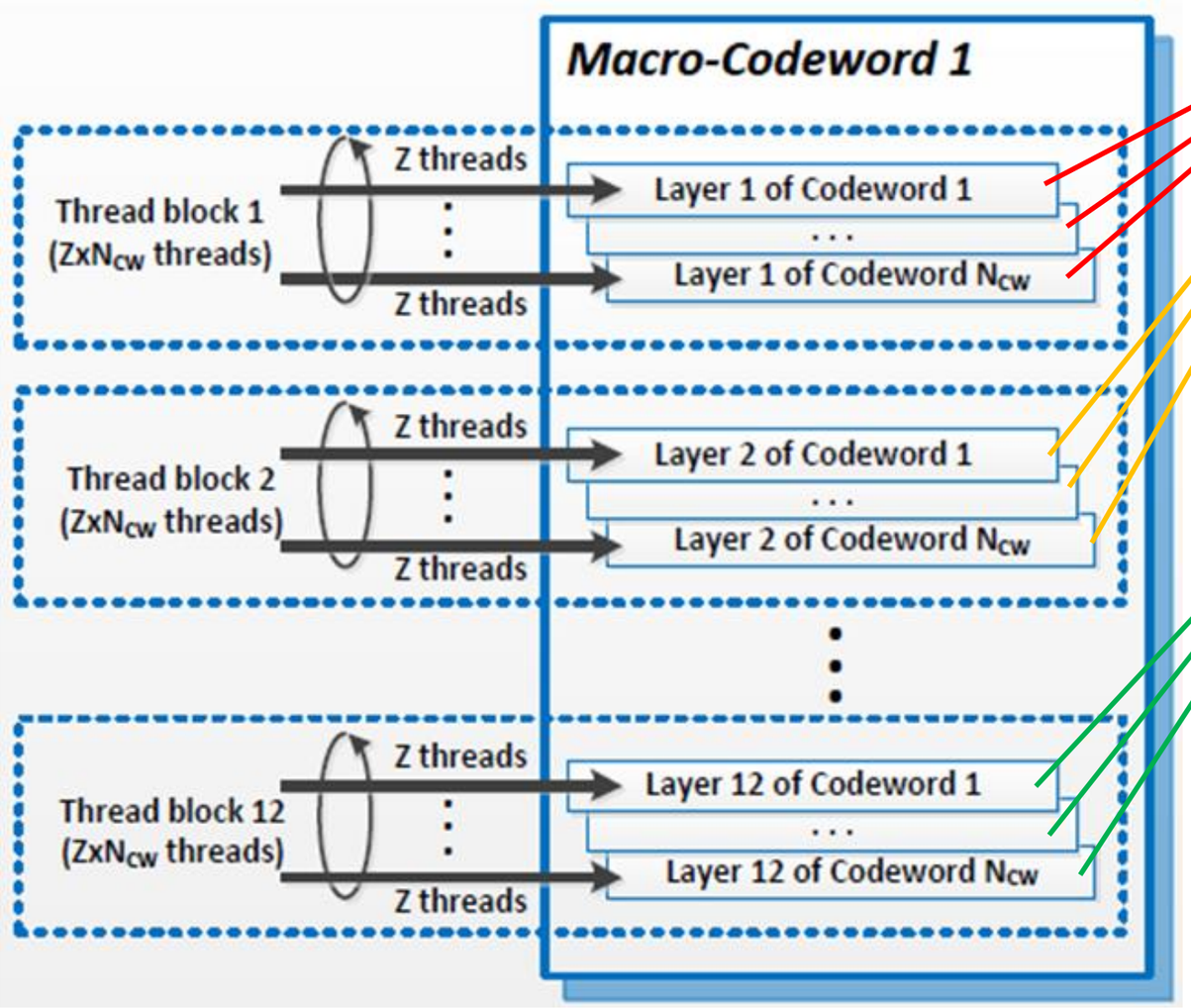
- Loosely coupled algorithm
  - Don t store q_{mn} in the memory.
  - Before computing $r_{mn,}$ recover q_{mn} first.
  - Good for CUDA implementation
    - Reduce the device memory storage
    - Reduce number of memory operations

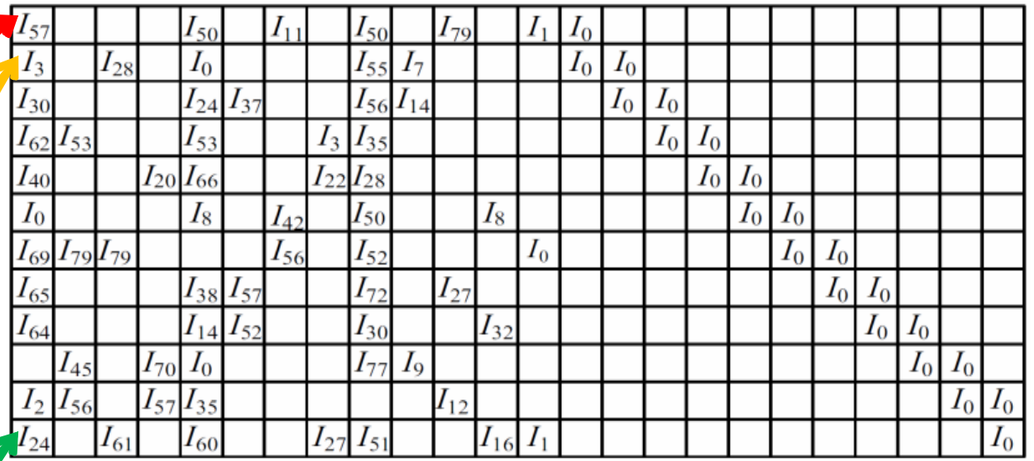


- Forward-backward check node update
  - For one row with ω_r non-zero element, we need to traverse the row for ω_r times.
  - Use forward-backward algorithms
  - **Reduce number of operations: M ωr(ωr-2) → M (3ωr-2)**
    - For example, M=2000, ωr=7, reduce ~50% operations.



Optimization: multi-codeword decoding

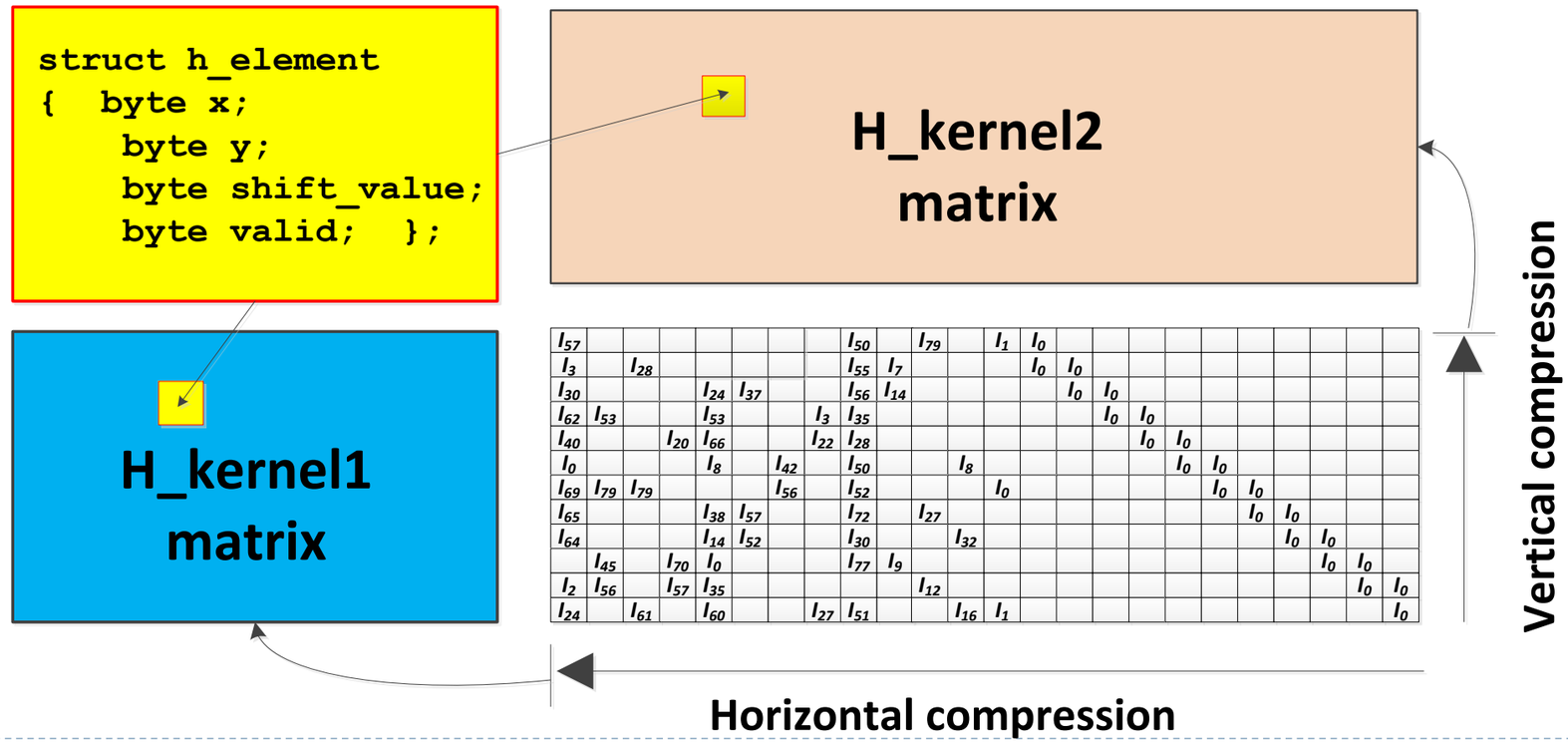




- Utilize the 2-D thread block structure
- Reduce diverse branches
- Take advantage of constant memory
- Good flexibility and scalability

Optimization – Efficient Storage

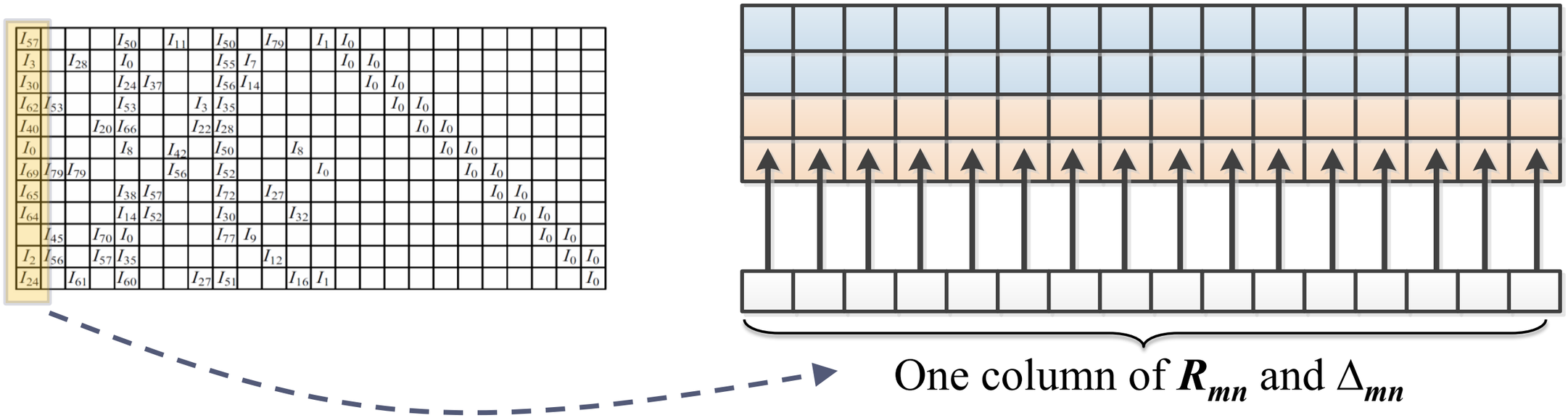
- Memory optimization
  - **Constant memory**: increase throughput by **8%**
  - **Compact representation**



Optimization – Memory Coalescing

- Coalescing device memory access
  - Compact format of $\boldsymbol{R}_{mn}$ and Δ_{mn} (check node message)
  - Writing compressed $\boldsymbol{R}_{mn}$ and Δ_{mn} matrices column-wise

  → coalesced memory access **(20% throughput improvement)**



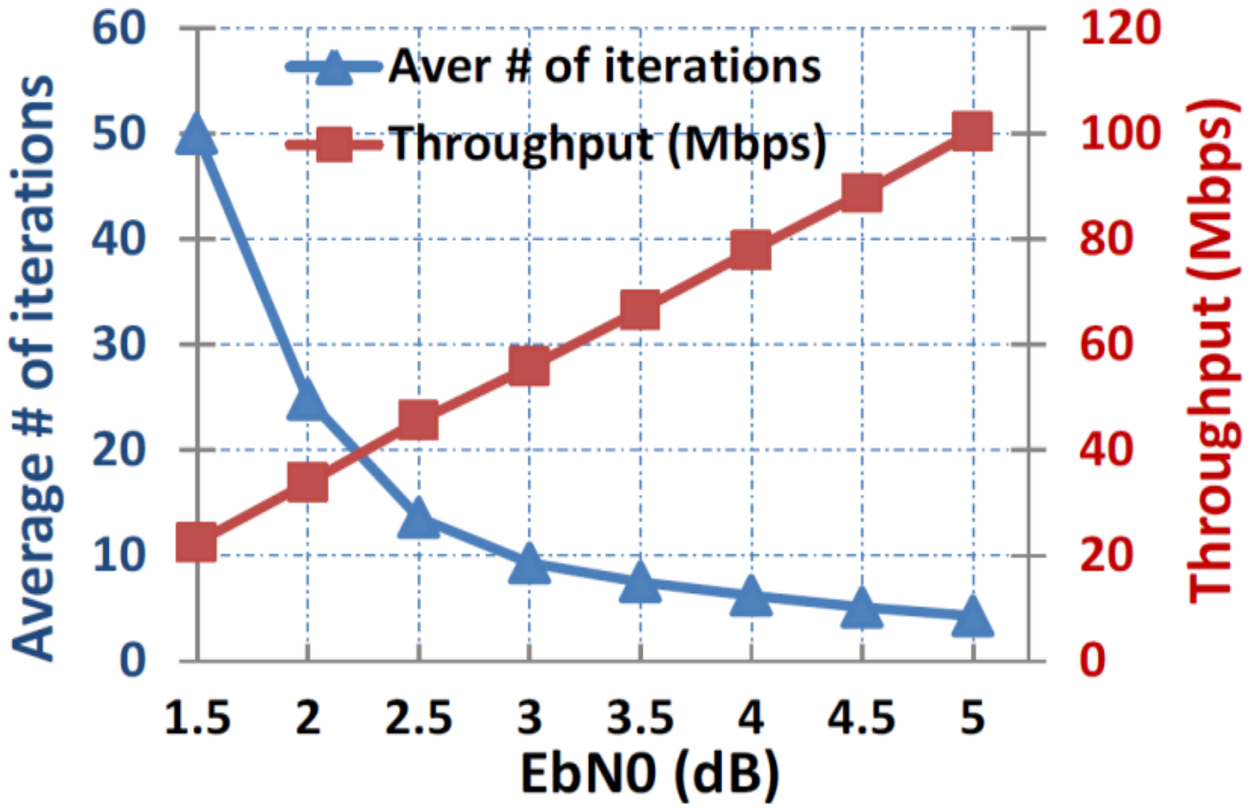
Experimental Results: LDPC Decoding Throughput

Code type	# of iterations	Decoding Time (ms)	Decoding Throughput (Mbps)
802.11n WiFi N=1944	5	26.0	74.7
	10	49.8	39.0
	15	71.5	27.2
802.16m WiMAX N=2304	5	24.0	96.1
	10	43.0	52.31
	15	64.0	36

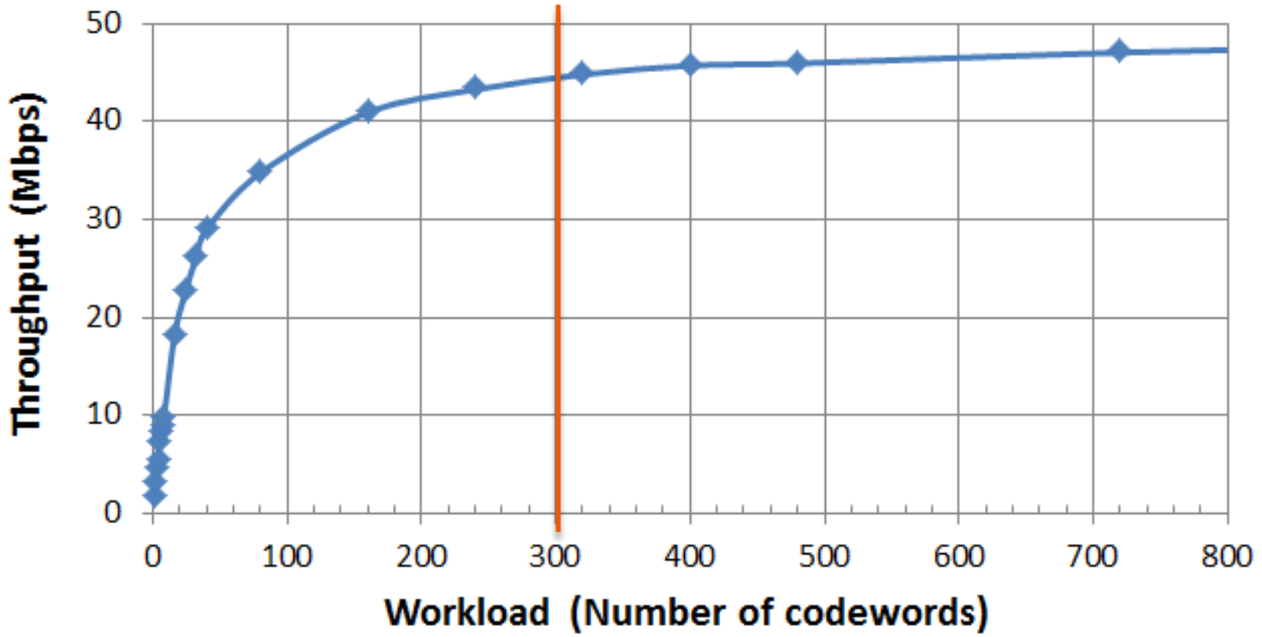
**** Host PC: Intel i5-750 Quad-core CPU, @2.67GHz, 8GB DDR3 memory***
**** GTX 470 Fermi GPU***

Experiment results: Early Termination Throughput

- Adaptive ET scheme:
  - Low SNR: ET off
  - High SNR: ET on
- Increase throughput for high SNR



Throughput VS Workload



- WiMax code, 2304 bits, rate ½ code
- At first, throughput increases almost linearly as workload increases
- After certain point, throughput stops increasing, because the threads occupies all the computation SMs in the GPU.

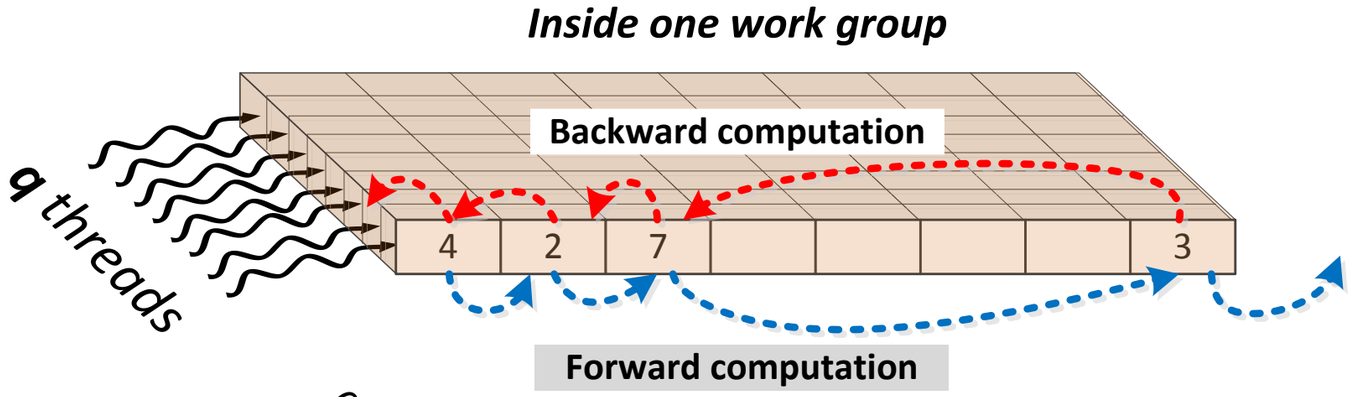
Comparison with Recently Published Work

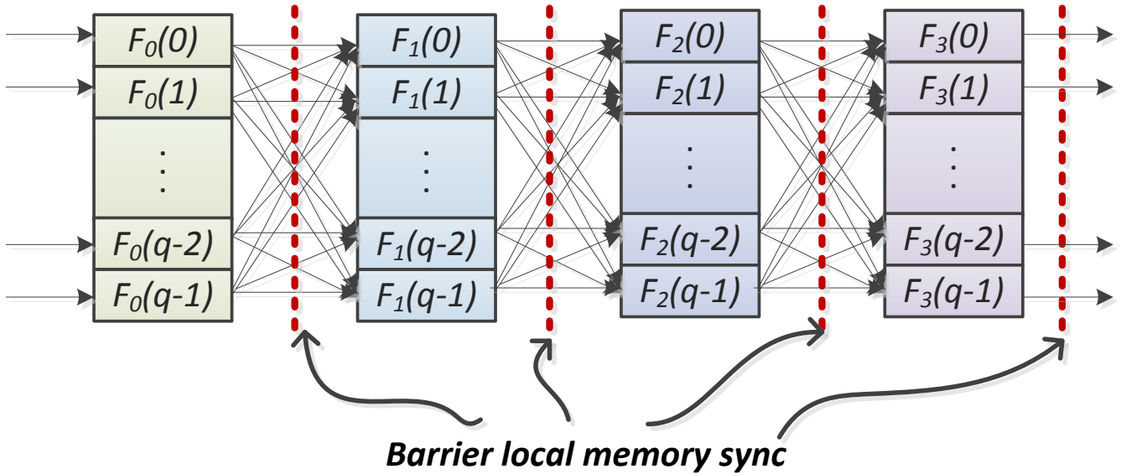
Work	Code length	Normalized throughput *(# of iterations = 10)*
Park et al, [Journal on WCN 2011]	18000 bits	2.809 Mbps
Yau et al, [ICACT 2011]	1/2 CMMB codes 9126 bits	2.046 Mbps
Zhao et al, [ICA3PP 2011]	4058 bits QC-LDPC	1.067 Mbps
Abburi, [VLSID 2011]	2034 bits WiMax	40 Mbps
Kennedy, [journal on WCN 2012]	2034 bits WiMax	32.9 Mbps
Kang [ICC 2012]	2048 bits, R=0.89	24.09 Mbps
Our work (*Results on GTX 470***) ***	**2304 bits WiMax**	**52.31 Mbps**

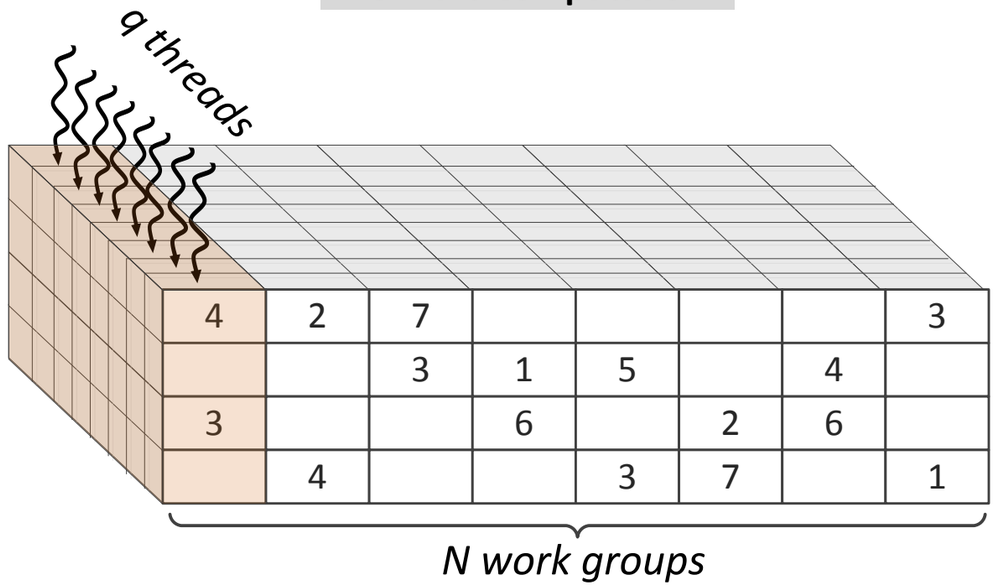
* G. Wang et al, ***A Massively Parallel Implementation of QC-LDPC Decoder on GPU***, *IEEE SASP 2011.*

Beyond Binary LDPC Codes – *GF(q)* Nonbinary LDPC

$$H = \begin{bmatrix} 4 & 2 & 7 & & & & & 3 \\ & & 3 & 1 & 5 & & 4 & \\ 3 & & & 6 & & 2 & 6 & \\ & 4 & & & 3 & 7 & & 1 \end{bmatrix}$$







G.Wang et al, ***Parallel Nonbinary LDPC Decoding on GPU***, *ASILOMAR Conference 2012.*

Conclusion

- Massively parallel implementations of a MIMO detector and a LDPC decoder on GPU
  - Tailor your algorithm
  - Tweak algorithm to improve efficiency
- Results:
  - Achieve high throughput
  - Faster than Existing work
  - Very flexible, can be a good platform for SDR systems
- Future work
  - Improving performance on Kepler
  - GPU accelerated SDR systems
- Links
  - Guohui Wang: www.GuohuiWang.com
  - Michael Wu: http://www.ruf.rice.edu/~mbw2/

Acknowledgement

- Research supported by US National Science Foundation under grants CNS-1265332, ECCS-1232274, EECS-0925942 and CNS-0923479.

- Equipment donations generously provided by NVIDIA.