新世代錯誤更正碼技術 -
低密度奇偶校驗碼 及其 晶片實現

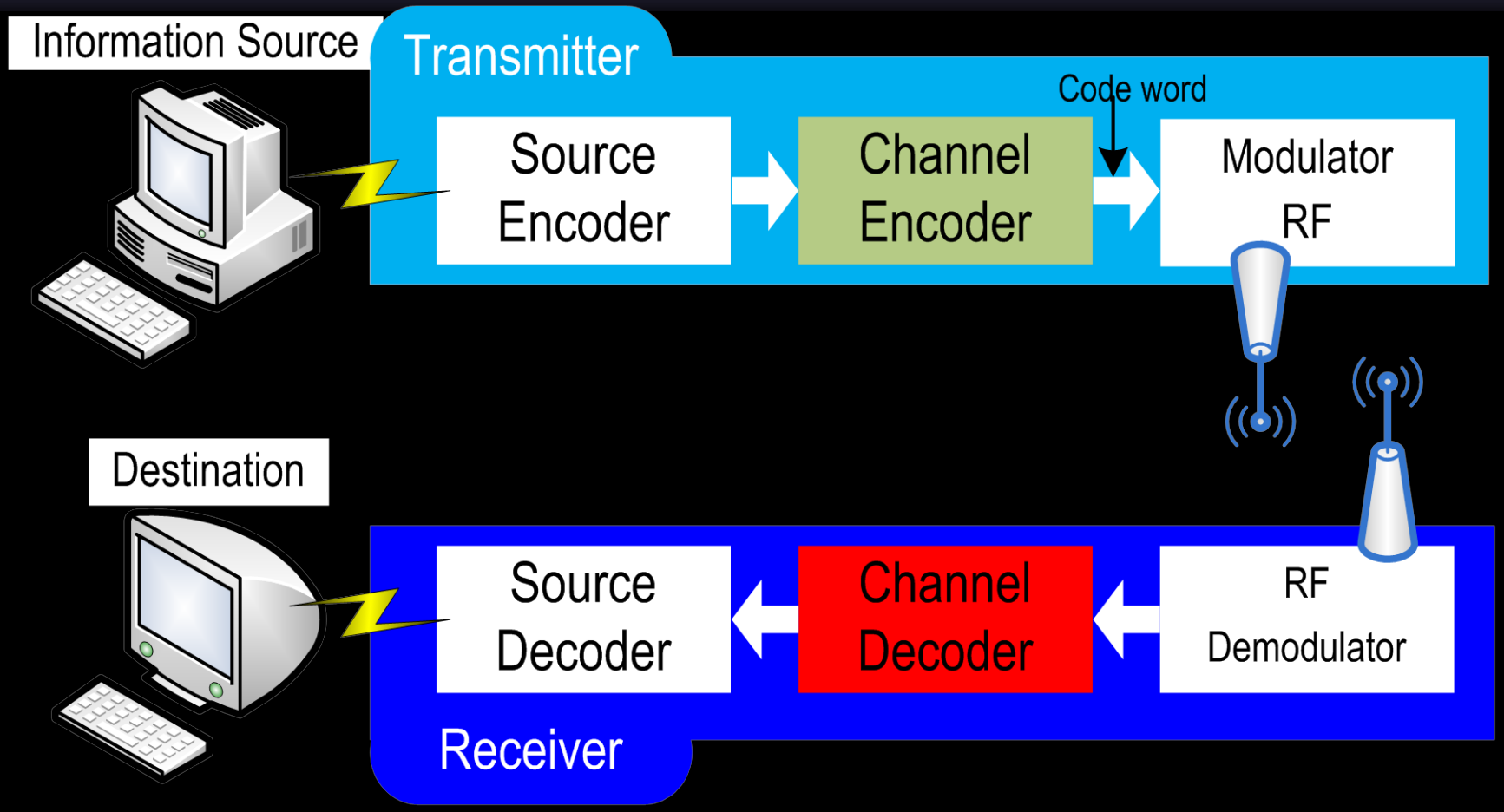
雲科大/電子系　楊博惠

2009/11/17

Outline

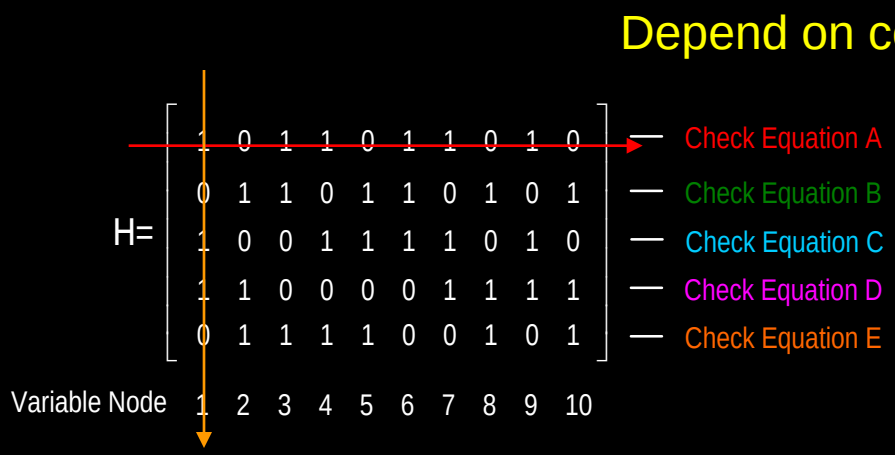
- Introduction to Low Density Parity Check (LDPC) codes
- LDPC Decoding Algorithms & Architectures
- Low Hardware Cost Look-Up Table (LUT)
- Unified Two's Complement Based Calculation
- OR Operation for Check Node Update
- Hardware Implementations
- Conclusion

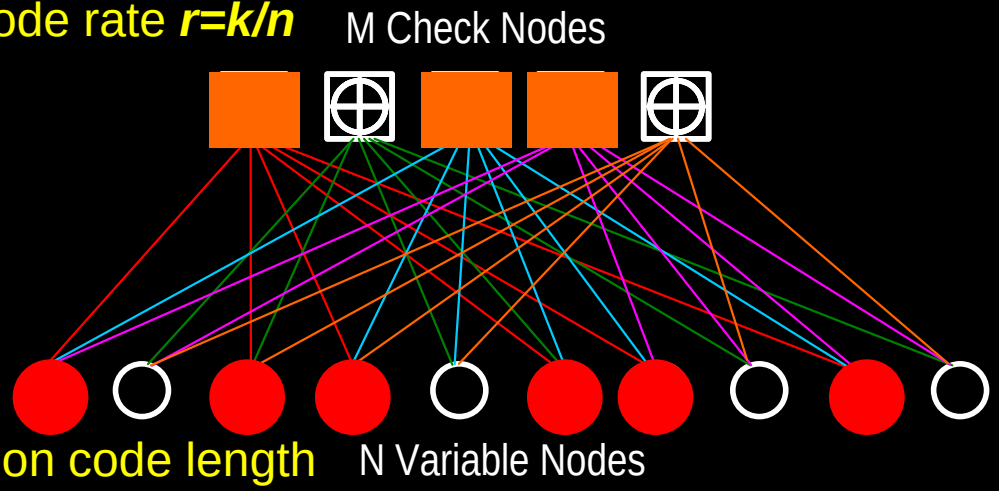
Digital Communication Block Diagram



Introduction of LDPC

- Low-density parity-check codes [1] proposed in 1962 are codes specified by a matrix containing mostly 0's and only a small number of 1's
- Use Tanner graph [2] representation LDPC codes
- A Tanner graph consists of two kinds of nodes, variable nodes and check nodes, and edges connecting them





Outline

- Introduction to Low Density Parity Check (LDPC) codes
- LDPC Decoding Algorithms & Architectures
- Low Hardware Cost Look-Up Table (LUT)
- Unified Two's Complement Based Calculation
- OR Operation for Check Node Update
- OR Operation with Pseudo-carry for Check Node Update
- Conclusion

LDPC Decoding Algorithm (1)

- The sum-product algorithm (SPA) is an iterative algorithm for decoding LDPC codes [1]

- Definition
  - *M(n)* : The set of Check Node connected to Variable Node
  - *N(m)* : The set of Variable Node connected to Check Node
  - *M(n)\n* : The set *M(n)* excluding the *n-th* Variable Node
  - *N(m)\m* : The set *N(m)* excluding the *m-th* Check Node
  - Log-Likelihood Ratio (LLR) is a soft information of the receiver

$$L(u_n) = \log \frac{P(u_n = 0 \mid y_n)}{P(u_n = 1 \mid y_n)} = L_c y_n$$

where u_n is the transmitted bit, y_n is the received bit, and $L_c = 2/\sigma^2$, where σ is the noise variance

LDPC Decoding Algorithm (2)

- Definition two LLR values
  - LLR of Variable Node to Check Node

$$\lambda_{n\to m}(u_n) = \log\left\{\frac{q_{n\to m}(0)}{q_{n\to m}(1)}\right\}$$

  - LLR of Check Node to Variable Node

$$\Lambda_{m\to n}(u_n) = \log\left\{\frac{r_{m\to n}(0)}{r_{m\to n}(1)}\right\}$$

$$\Lambda_1^{\text{out}} = \frac{P_{20}P_{31} + P_{21}P_{30}}{P_{20}P_{30} + P_{21}P_{31}}$$

- The iteration procedures of SPA are summarized as three follows:
  - Initialization

$$\lambda_{n\to m}(u_n) = L(u_n)$$

$$\Lambda_{m\to n}(u_n) = 0 \qquad (1)$$

LDPC Decoding Algorithm (3)

- Iterative process
  - Check Node Update:

$$\Lambda_{m\to n}(u_n) = \text{sign} \prod_{n'\in N(m)\backslash n} \lambda_{n'\to m}(u_{n'}/2) \cdot \phi^{-1}\left(\sum_{n'\in N(m)\backslash n} \phi\left[\lambda_{n'\to m}(u_{n'}/2)\right] \right) \qquad (2)$$

where $\phi(x) = \log\left(\frac{e^x+1}{e^x-1}\right)$ and $\phi^{-1}(x) = \phi(x)$

Because $\phi\left(\phi(x)\right) = x$

  - Approximated formula (2) and called Sign-Min Algorithm [3]

$$\Lambda_{m\to n}(u_n) = \text{sign} \prod_{n'\in N(m)\backslash n} \lambda_{n'\to m}(u_{n'}/2) \cdot \min \lambda_{n'\to m}(u_{n'}/2) \qquad (3)$$

LDPC Decoding Algorithm (4)

– Variable Node Update:

$$\lambda_{n\to m}(u_n) = L(u_n) + \sum_{m'\in M(n)\setminus m} \Lambda_{m'\to n}(u_n) \qquad (4$$

- Hard Decision

$$\lambda_n(u_n) = L(u_n) + \sum_{m\in M(n)} \Lambda_{m\to n}(u_n) \qquad (5$$

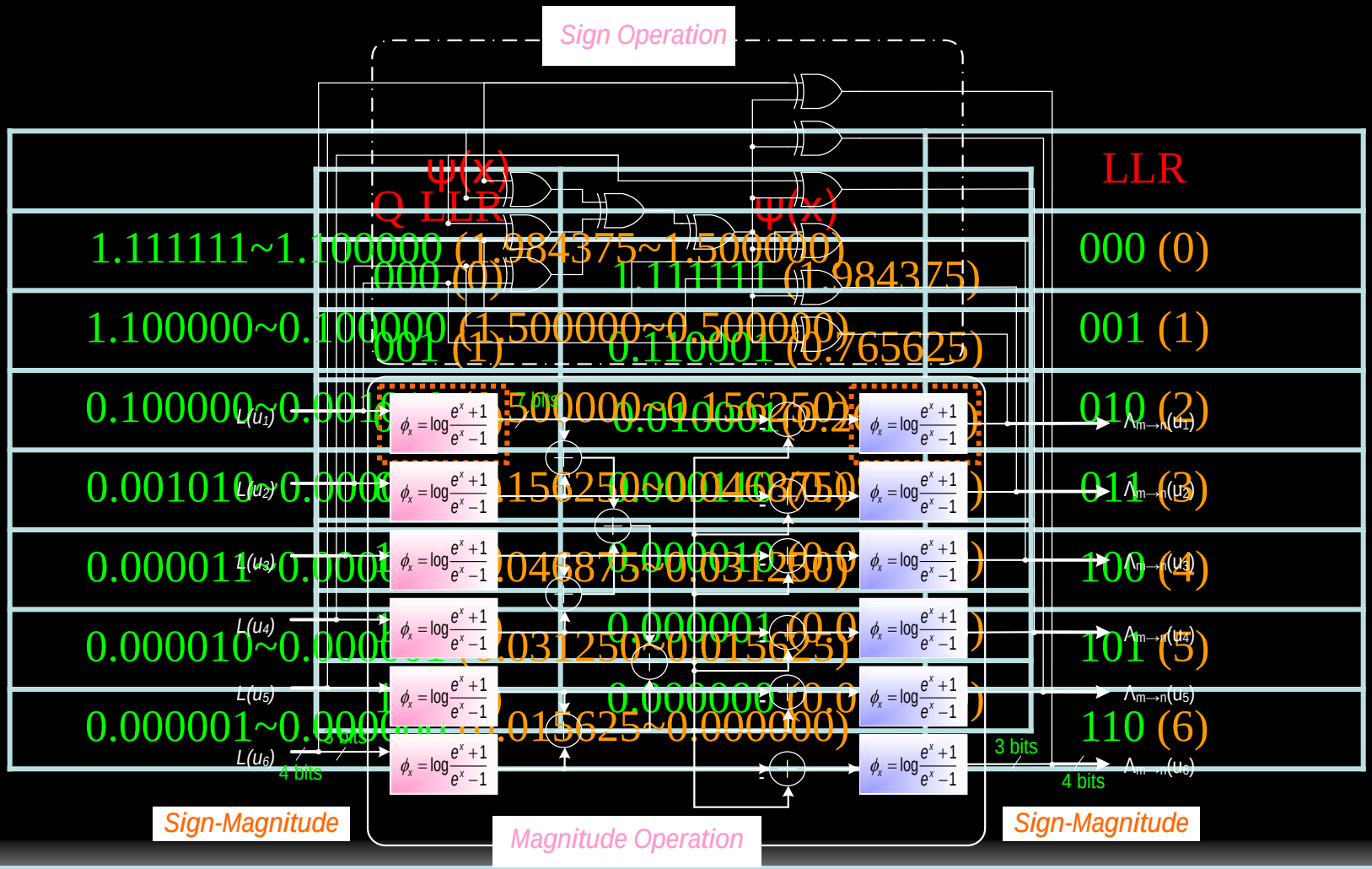
$$\begin{cases} \hat{u}_n = 0 & if \quad \lambda_n(u_n) \geq 0 \\ \hat{u}_n = 1 & if \quad \lambda_n(u_n) < 0 \end{cases}$$

– After each iteration, a hard decision is made

– Compute the matrix $\hat{u}_n H^T$, if $\hat{u}_n H^T$ =0 or get up to maximum the numbers of the decoding iterations, then halt the algorithm. Otherwise, the algorithm repeats iterative process execute next iteration

Check Node Architecture [6]

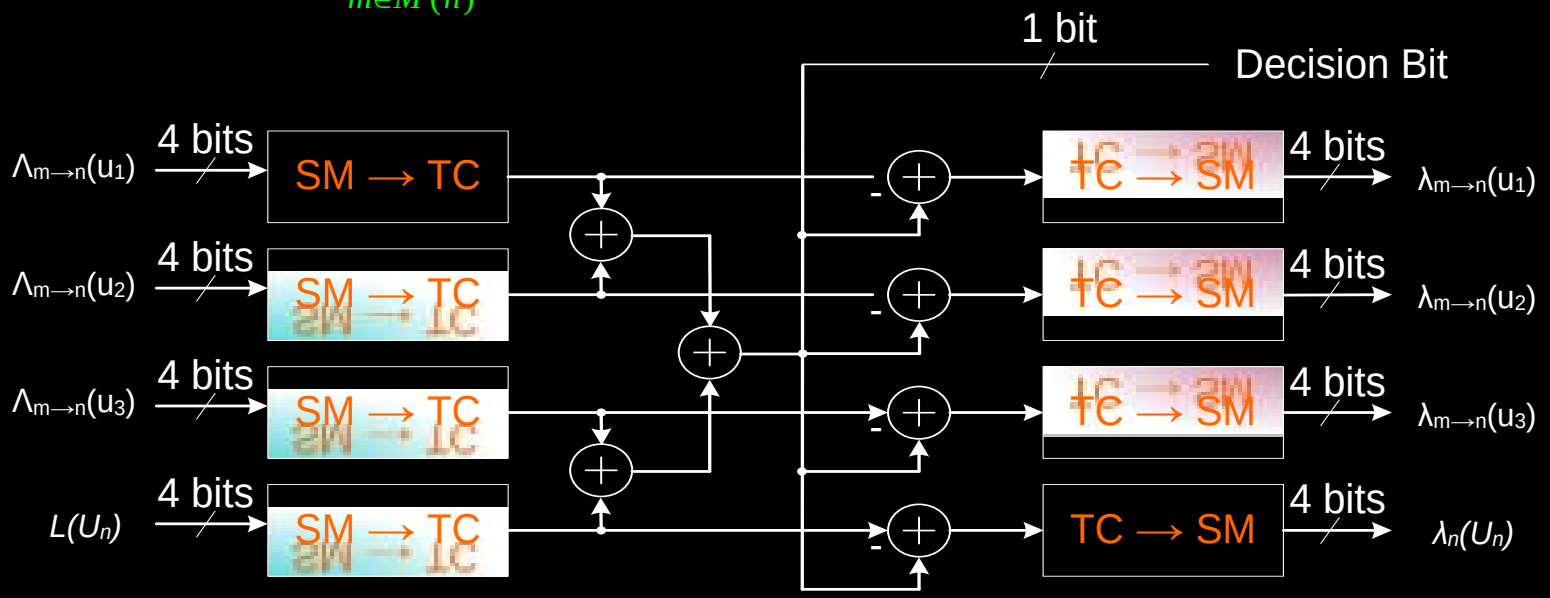
$$\Lambda_{m\to n}(u_n) = \text{sign} \prod_{n'\in N(m)\setminus n} \lambda_{n'\to m}(u_{n'}/2) \cdot \phi^{-1}\left(\sum_{n'\in N(m)\setminus n} \phi\left[\lambda_{n'\to m}(u_{n'}/2)\right] \right)$$



Variable Node Architecture [6]

$$\lambda_{n\to m}(u_n) = L(u_n) + \sum_{m'\in M(n)\setminus m} \Lambda_{m'\to n}(u_n)$$

$$\lambda_n(u_n) = L(u_n) + \sum_{m\in M(n)} \Lambda_{m\to n}(u_n)$$

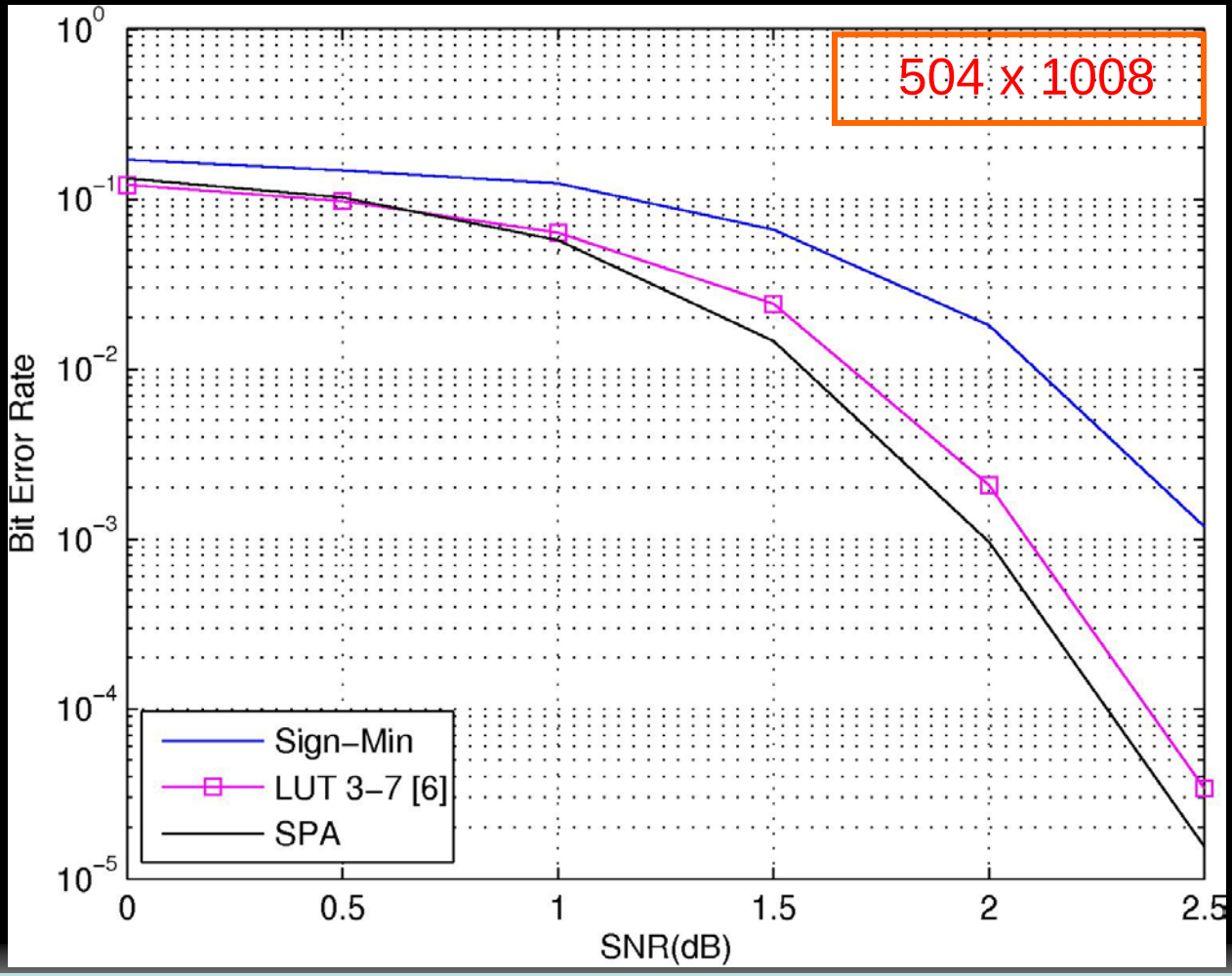


SM : Sign-Magnitude

TC: Two's Complement

BER Simulation Result

- Use *504x1008* matrix [4] and 80 decoding iterations [5]

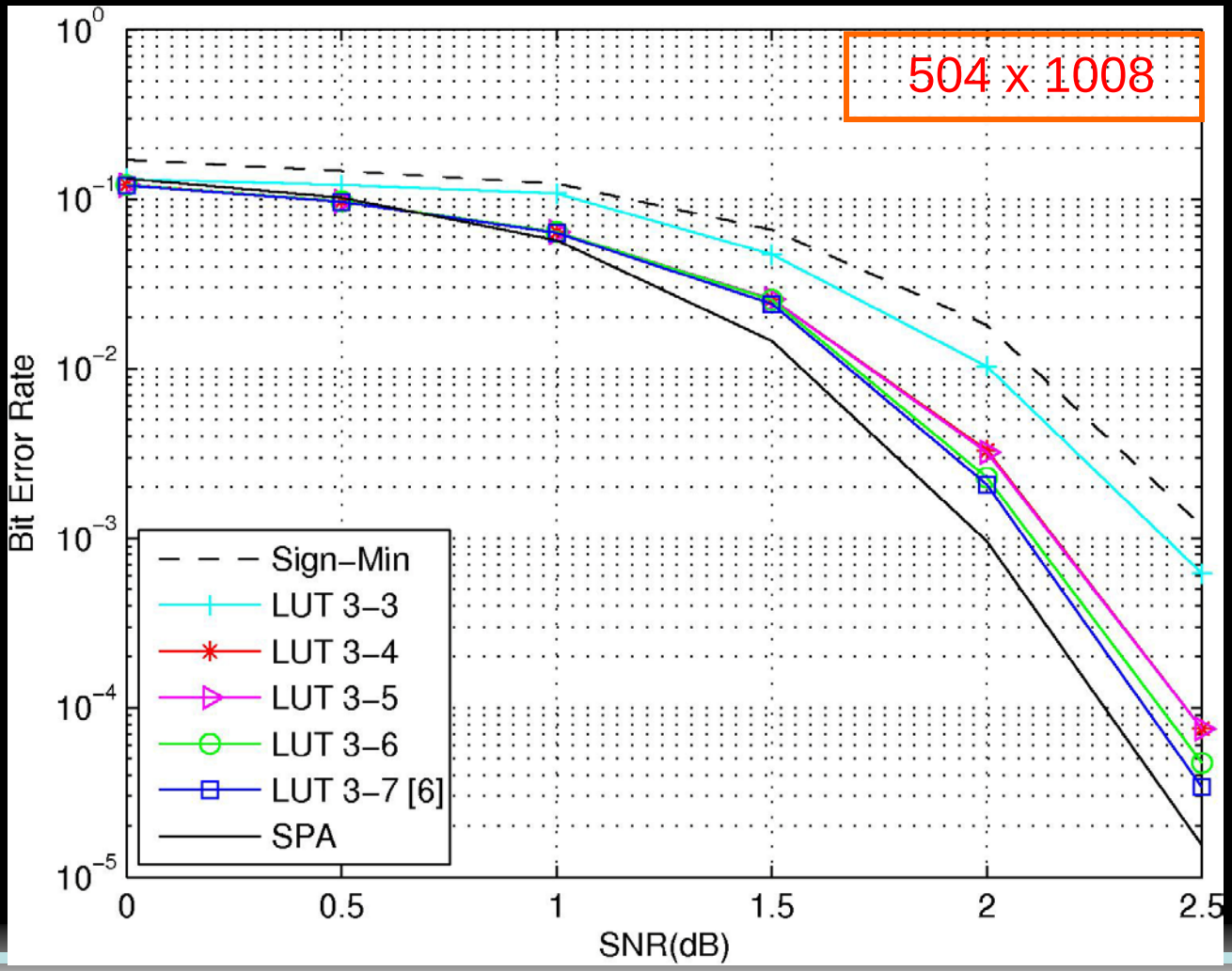


Outline

- Introduction to Low Density Parity Check (LDPC) codes
- LDPC Decoding Algorithms & Architectures
- Low Hardware Cost Look-Up Table (LUT)
- Unified Two's Complement Based Calculation
- OR Operation for Check Node Update
- Hardware Implementations
- Conclusion

Different Bits of LUT Simulation Result

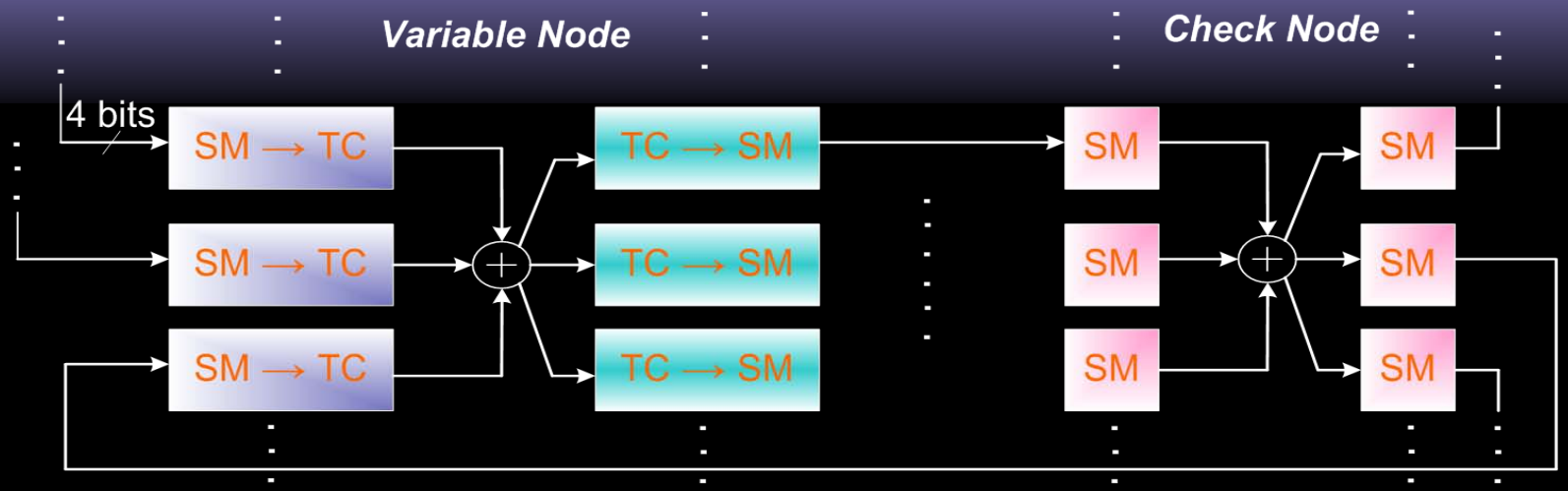
- Use *504x1008* matrix [4] and 80 decoding iterations [5]

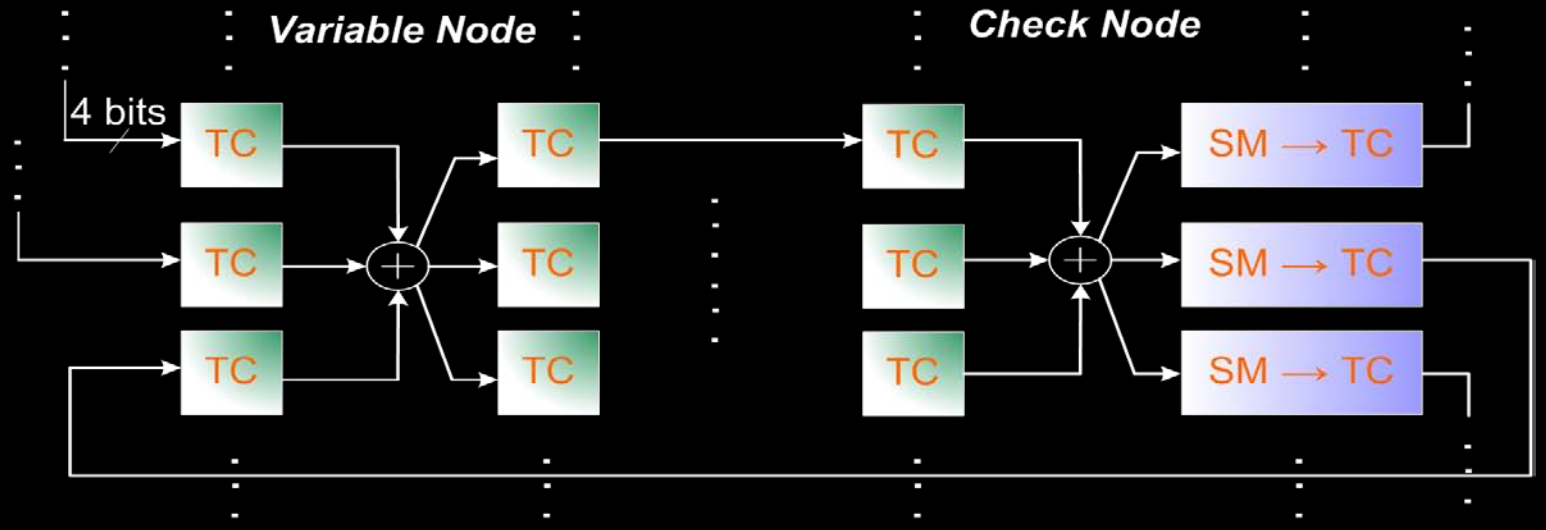


Outline

- Introduction to Low Density Parity Check (LDPC) codes
- LDPC Decoding Algorithms & Architectures
- Low Hardware Cost Look-Up Table (LUT)
- Unified Two's Complement Based Calculation
- OR Operation for Check Node Update
- Hardware Implementations
- Conclusion

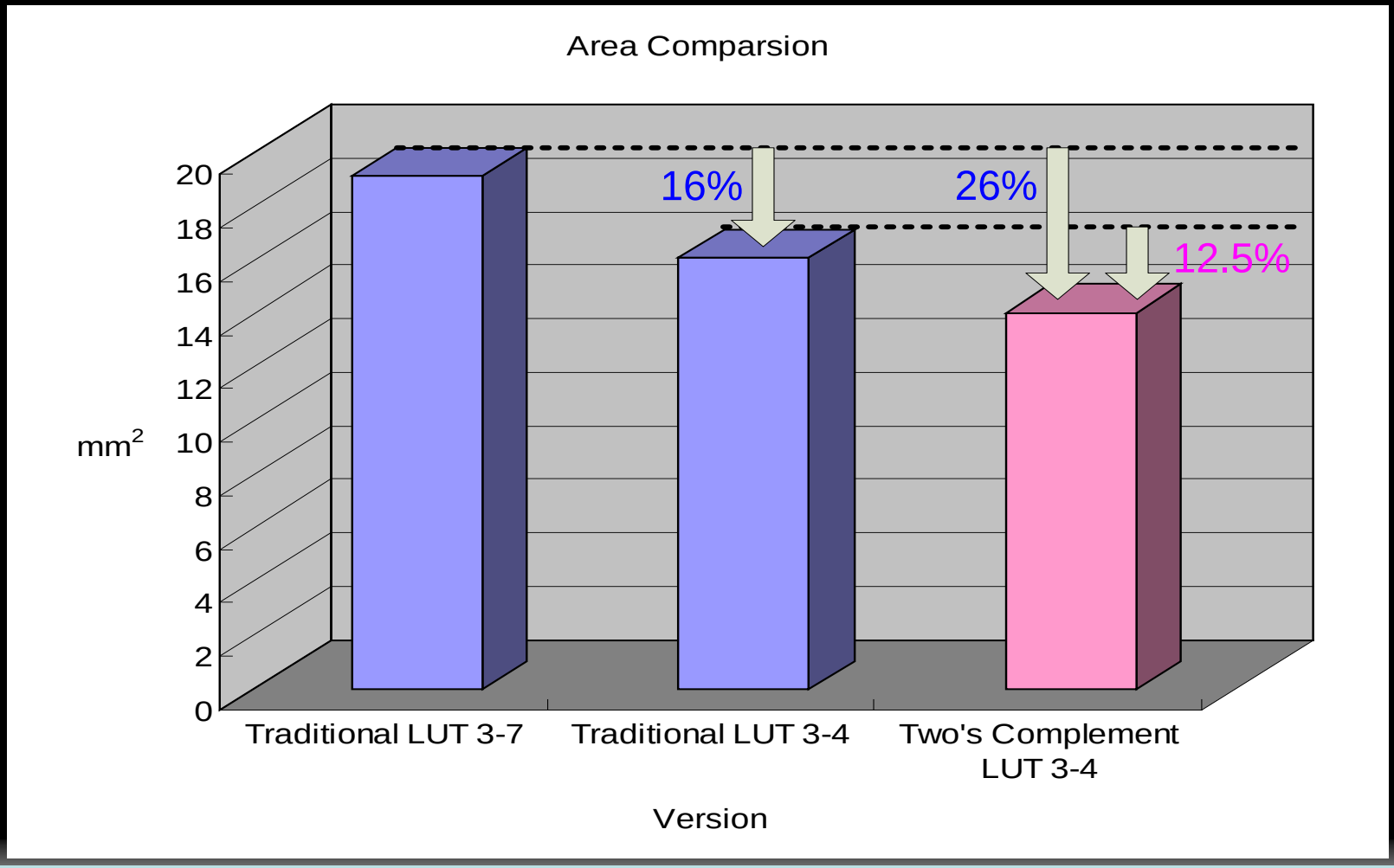
Traditional Update vs 2's Complement Update





Area Comparison

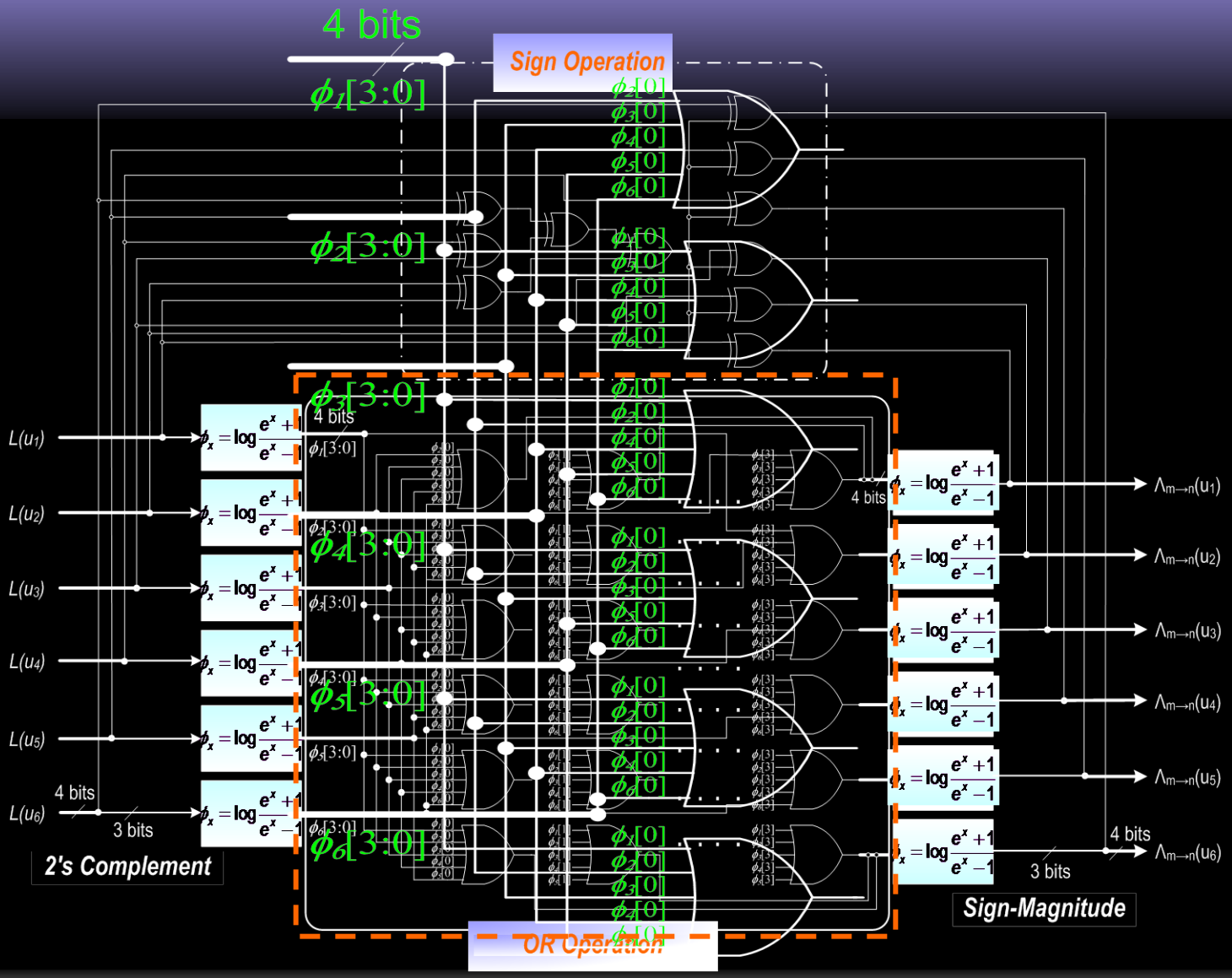
Synthesized by TSMC 0.18μm Artisan Standard Cell Library



Outline

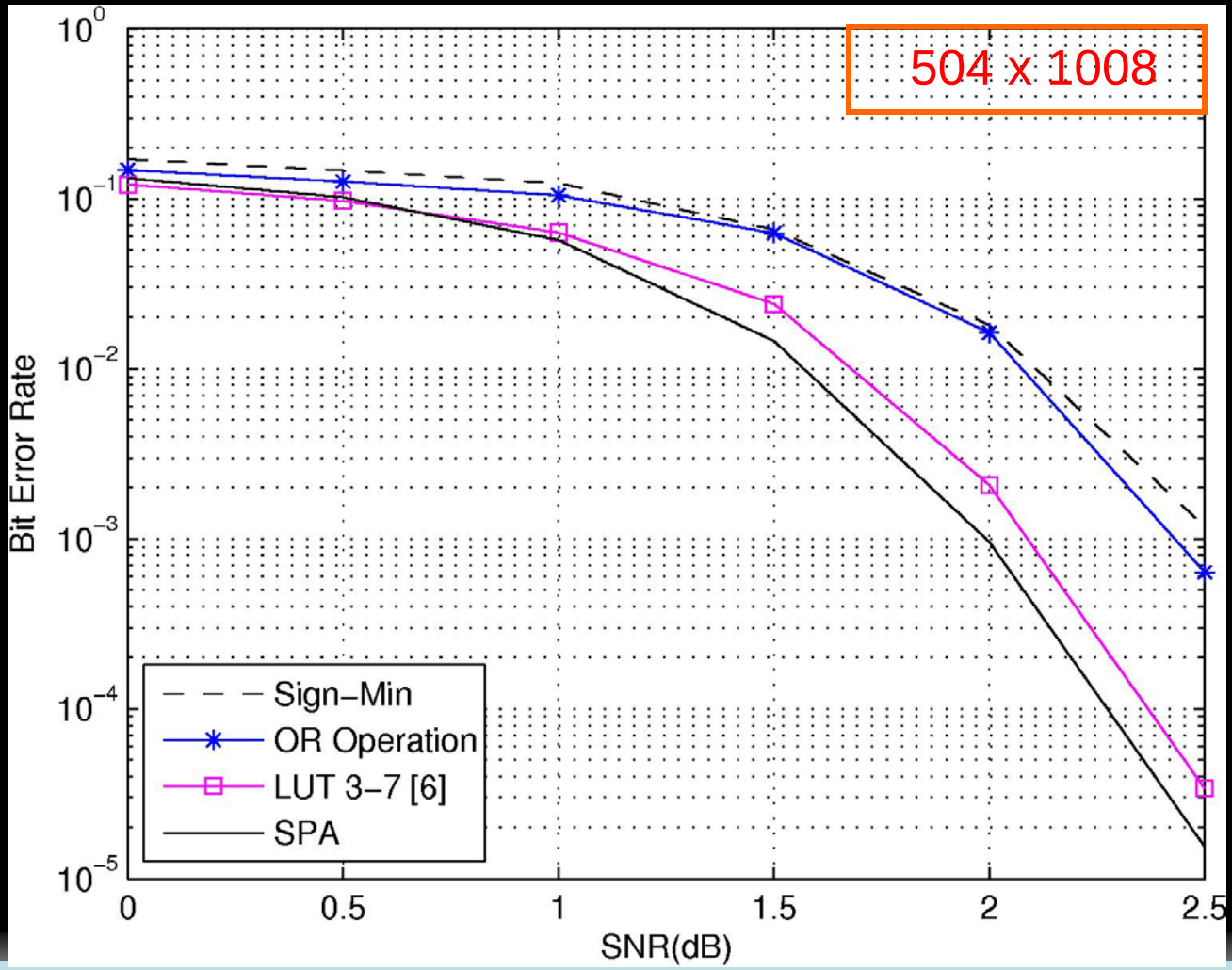
- Introduction to Low Density Parity Check (LDPC) codes
- LDPC Decoding Algorithms & Architectures
- Low Hardware Cost Look-Up Table (LUT)
- Unified Two's Complement Based Calculation
- OR Operation for Check Node Update
- Hardware Implementations
- Conclusion

OR operation for Check Node Architecture

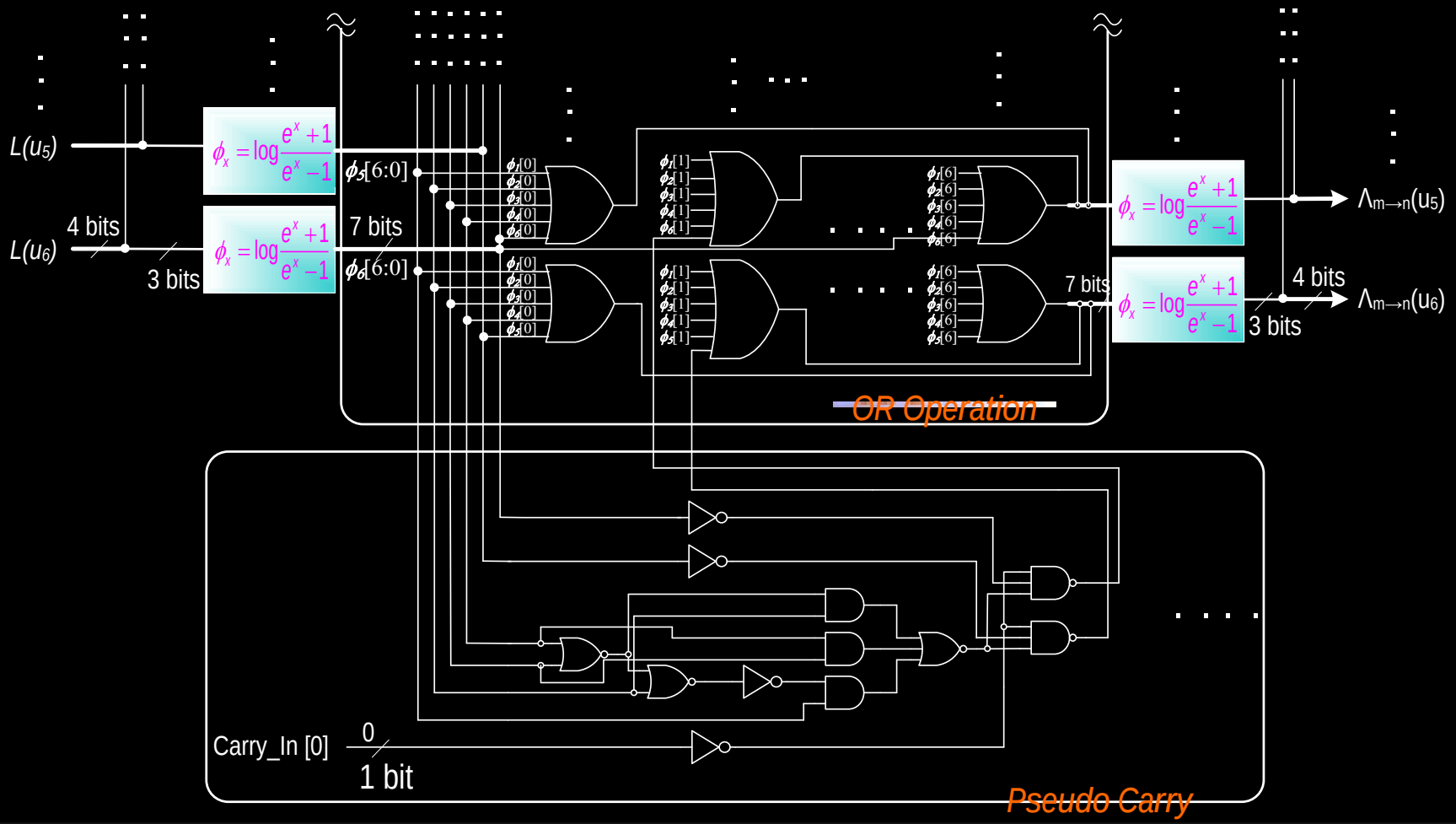


OR Operation Simulation Result

- Use *504x1008* matrix [4] and 80 decoding iterations [5]

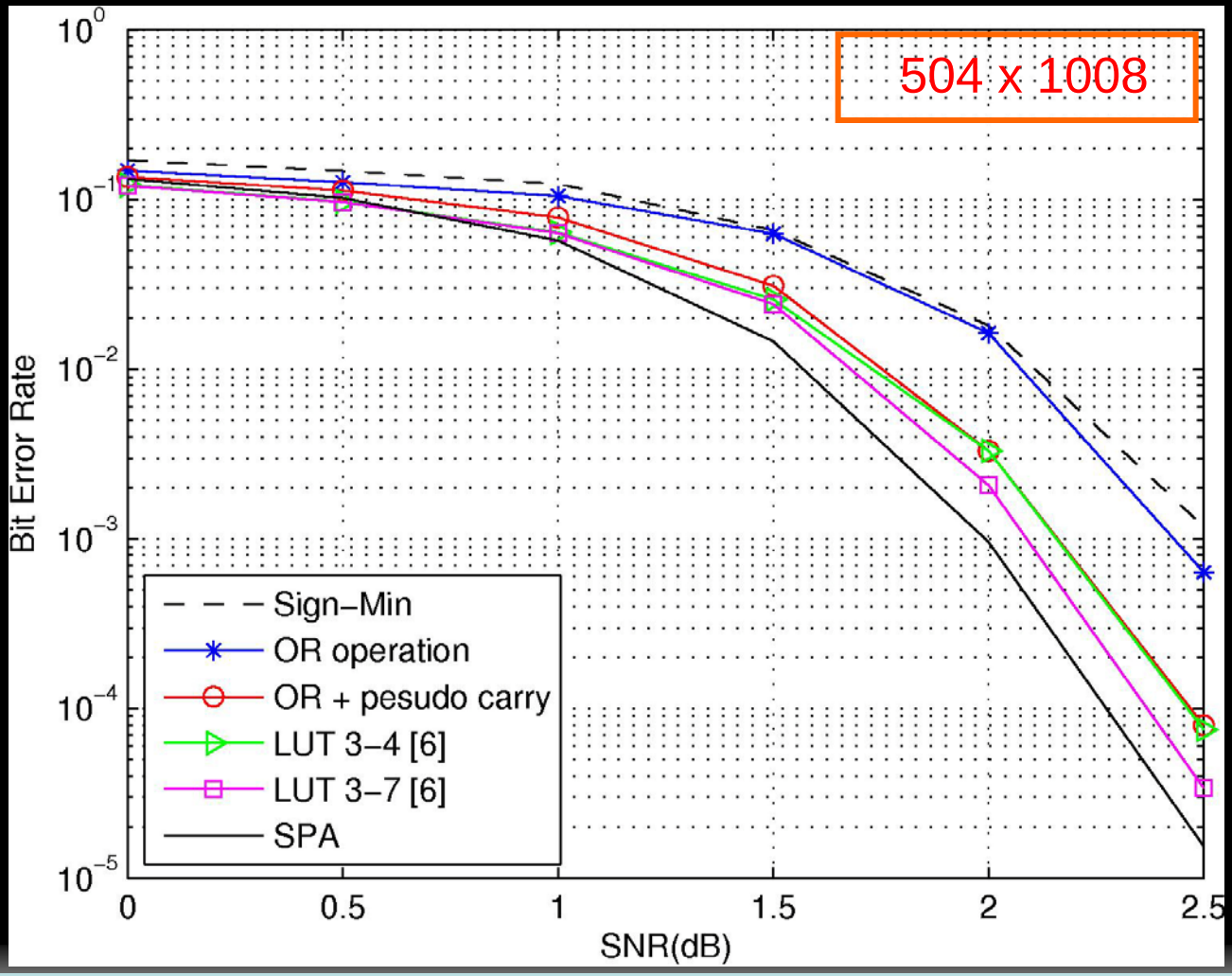


OR Operation with Pseudo-carry for Check Node Architecture



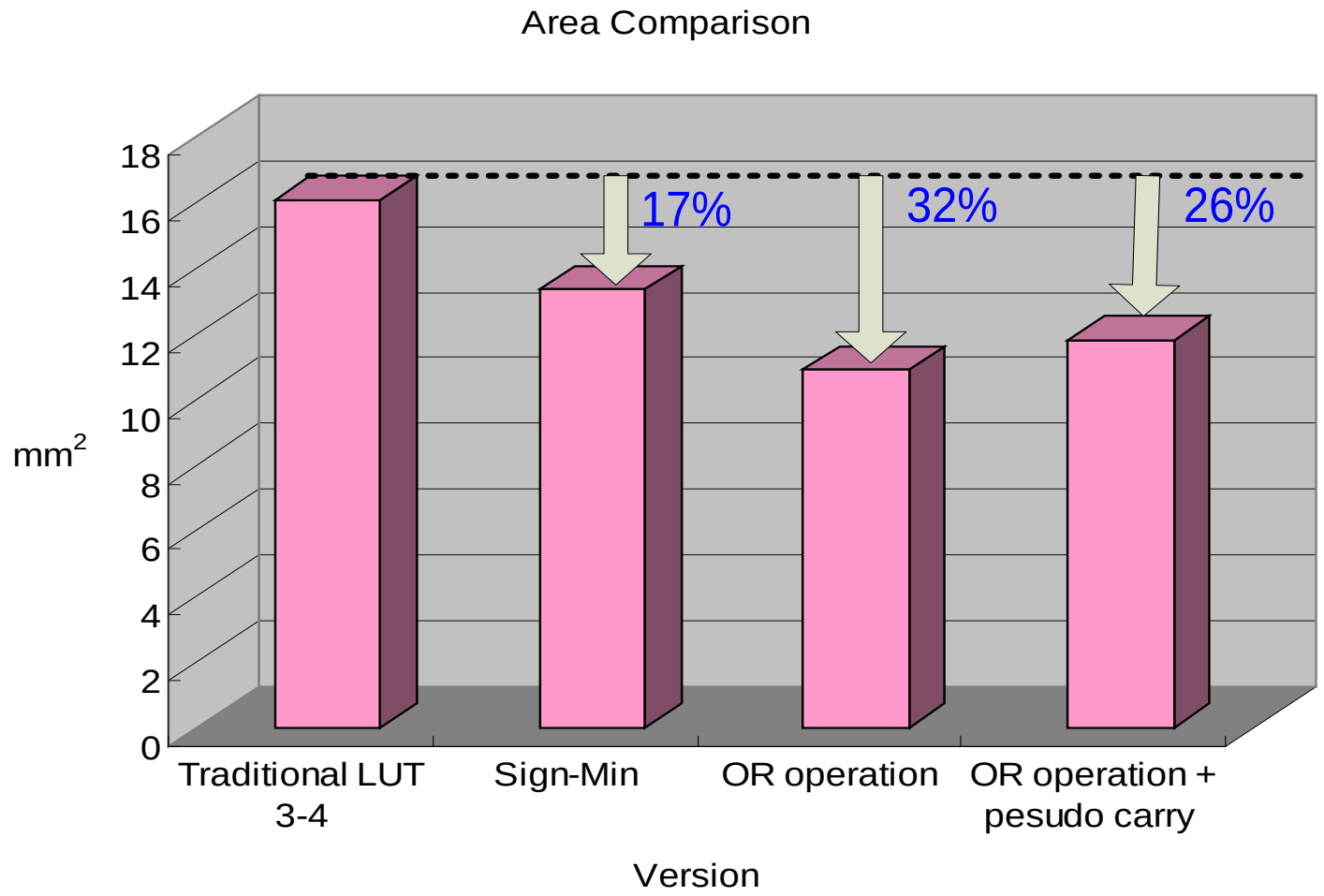
OR Operation with Pseudo-carry Simulation Result

- Use *504x1008* matrix [4] and 80 decoding iterations [5]



Area Comparison

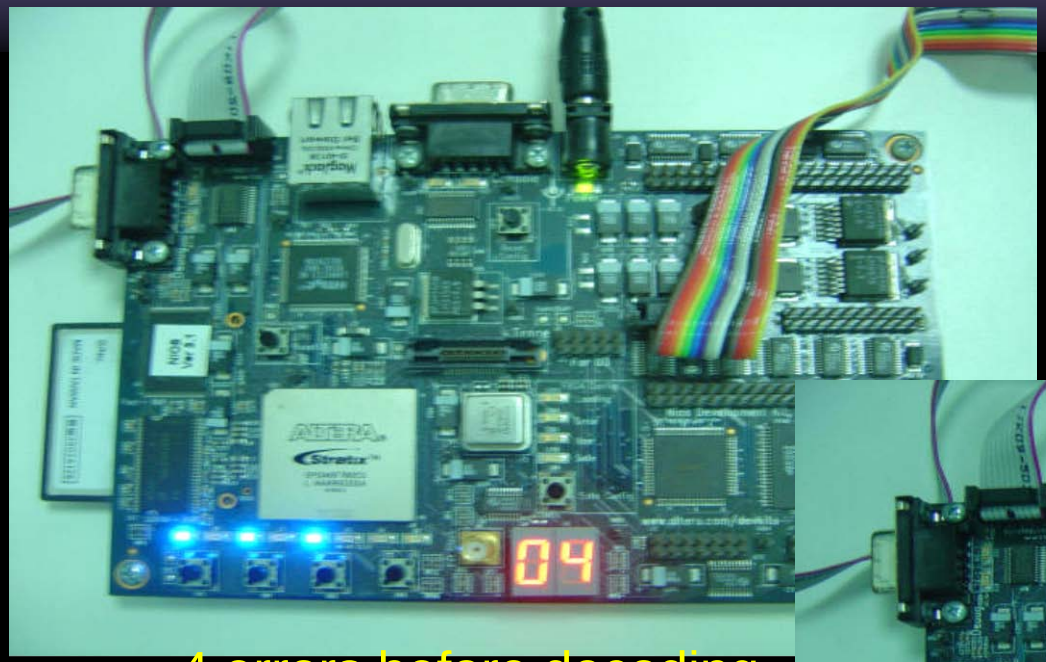
Synthesized by TSMC 0.18μm Artisan Standard Cell Library



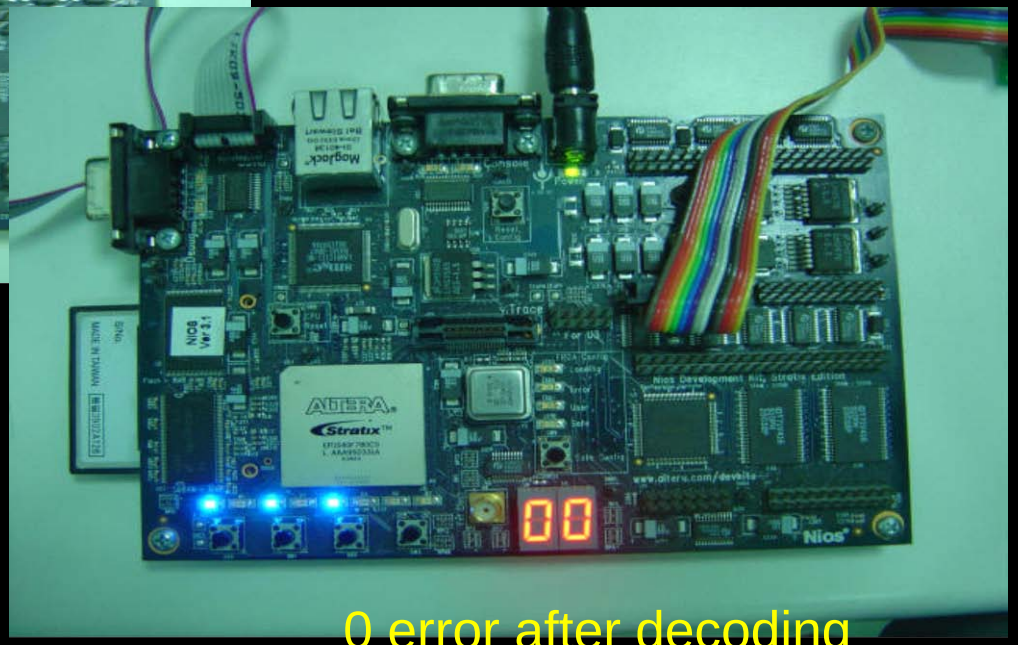
Outline

- Introduction to Low Density Parity Check (LDPC) codes
- LDPC Decoding Algorithms & Architectures
- Low Hardware Cost Look-Up Table (LUT)
- Unified Two's Complement Based Calculation
- OR Operation for Check Node Update
- Hardware Implementations
- Conclusion

FPGA Implementation



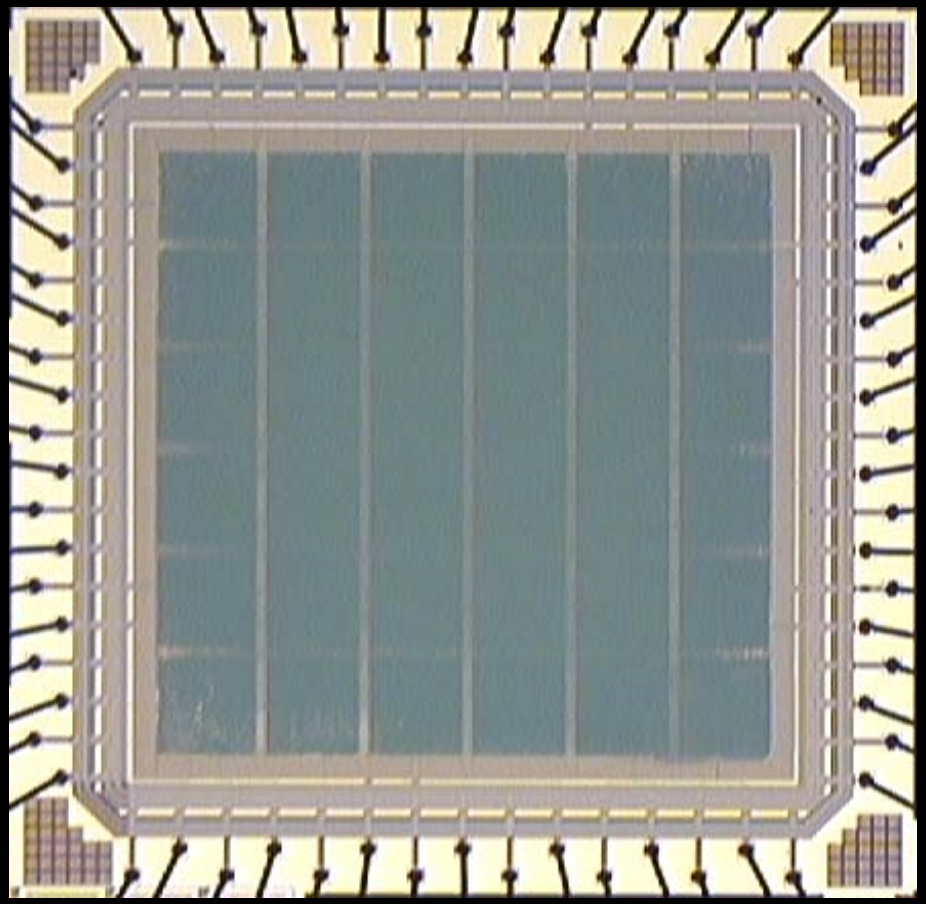
4 errors before decoding



0 error after decoding

Implemented Chip

- TSMC 0.18μm Artisan Standard Cell Library



Technology	TSMC 0.18μm
Work Voltage	1.8 / 3.3 V
Core Size	2.8 x 2.8 ($7.84mm^2$)
Die Size	3.3 x 3.3 ($10.89mm^2$)
Frequency	50MHz
Throughput	24.57Mbps (max)
Power	89.28mW @50MHz

Outline

- Introduction to Low Density Parity Check (LDPC) codes
- LDPC Decoding Algorithms & Architectures
- Low Hardware Cost Look-Up Table (LUT)
- Unified Two's Complement Based Calculation
- OR Operation for Check Node Update
- Hardware Implementations
- Conclusion

Conclusion

- The LDPC is an excellent code, and it is widely used in communication standards.
- Algorithm study first, and then work out the hardware.
- The targets of the hardware are:
  - Smaller chip size
  - Lower power consumption
  - Higher operational speed
- Team work is the key to win.
- We worked out a good cooperate model in our laboratory.

Reference

- [1] R. G. Gallager, "Low-density parity-check code", *IRE Trans. Inform.Theory, vol.IT-8, pp. 21-28, Jan. 1962.*
- [2] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 5, pp. 533-547, 1981.
- [3] W. E. Ryan, An introduction to LDPC codes, in CRC Handbook for coding and signal processing for recording systems (B. Vasic ed.), CRC Press, 2004 .
- [4] D. J. C. Mackay, *Online database of low-density parity-check codes*, available at http://www.inference.phy.cam.ac.uk/mackay/CodesFiles.html.
- [5] S. Tong, P. Wang, D. Wang and X. Wang, " Box-minus operation and application in sum-product algorithm", *IEE Electronics Letters, vol. 41, pp. 197-198, Feb. 2005.*
- [6] A. J. Blanksby and C. J. Howland, "A 690-mW 1 Gb/s 1024-b rate-1/2 low-density parity-check code decoder", *IEEE J. Solid-State Circuits, vol. 37, no. 3, pp. 404–412, Mar. 2002.*