Generic model of LDPC code decoders

Frédéric GUILLOUD
Emmanuel BOUTILLON
guilloud@enst-bretagne.fr
emmanuel.boutillon@univ-ubs.fr

OUTLINE

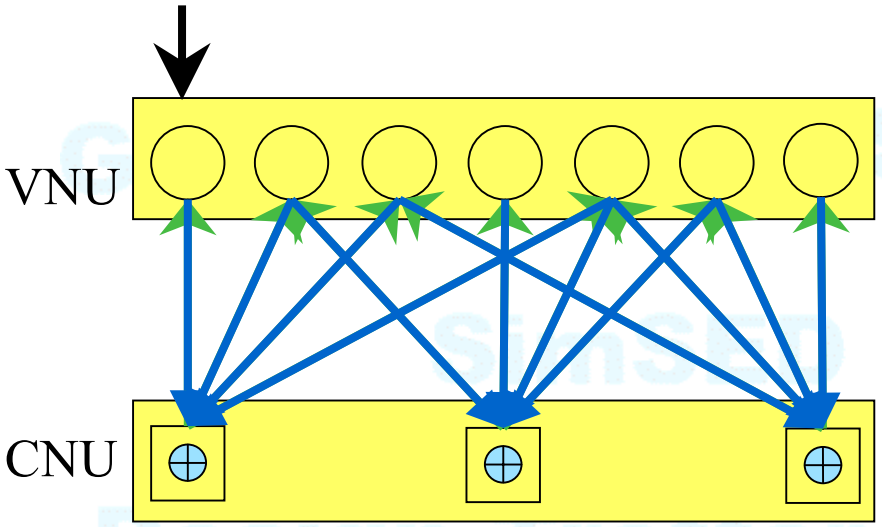
- Motivation
- Architecture parameters
- Example of architecture
- Conclusion

Motivation

- Find « generic parameters » to characterize a LDPC architecture.
- (Try to) classify existing architectures
- Propose new architectures

Hypothesis 2

- Regular (j,k) LDPC code
- Believe Propagation decoding algorithm
- Check node in the frequency domain



$$E_{m,n} = I_n + \sum_{m' \in M(n) \setminus m} T_{m',n}$$

$$T_{m,n} = \Phi^{-1}\left(\sum_{n' \in N(m) \setminus n} \Phi(E_{m,n'}) \right)$$

$$\Phi(x) = -\ln\left(\tanh\left(x/2\right)\right)$$

OUTLINE

- Motivation
- Architecture parameters
  - Parallelism
  - Scheduling
  - Type of generic node (VNU, GNU)
  - Position of the interconnection network
- Example of architecture
- Conclusion

Parallelism of a $(j,k)=(3,6)$ LDPC

Pc : computational power: number of edges computed/cycle

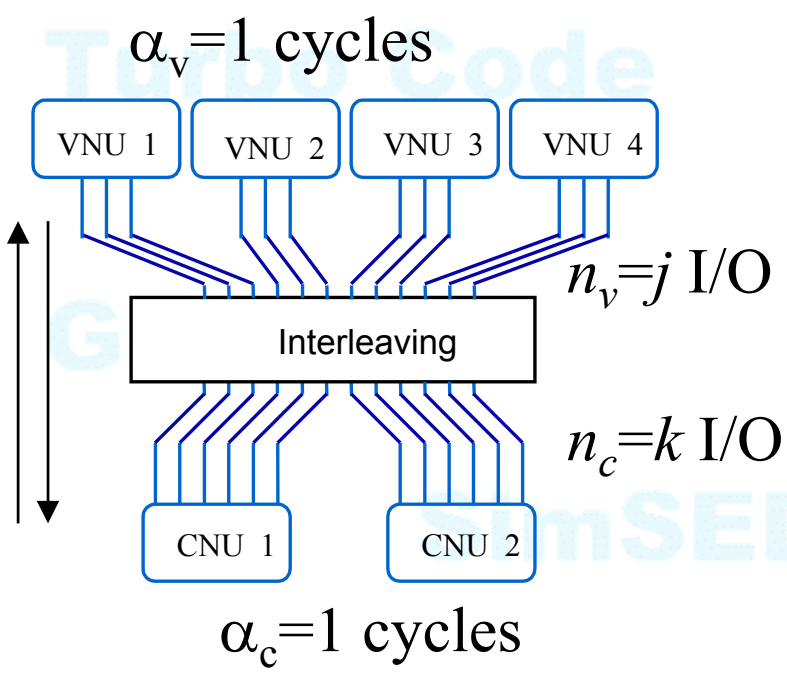
K number of information bit :	K
R rate of the code	$R=(1-j/k)$
E number of edge of the code:	jK/R
F_{clock} Clock frequency:	100 Mbits/s
F_{bit} Bit throughput :	10 Mbit/s
N_{it} average number of iteration :	20

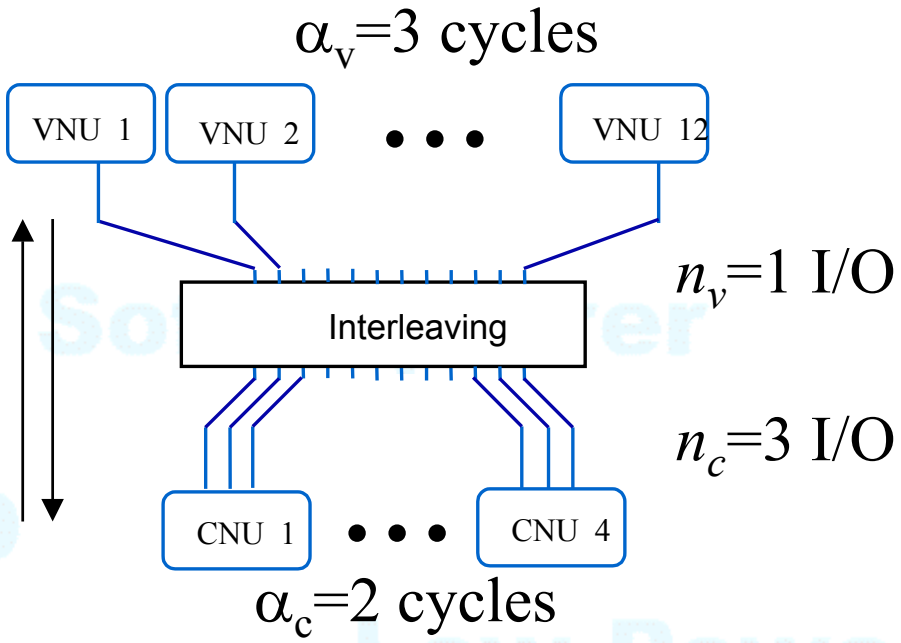
$$P_c = EN_{it}\ [\text{edge/codeword}] \times \left(\frac{F_{bit}}{K}\right) \frac{1}{F_{clock}}\ [\text{codeword/clock cycle}]$$

$$= \frac{3K}{0.5} \times 20 \times \frac{10}{K} \times \frac{1}{100} = 12[\text{edge/clock cycle}]$$

Serial/Parallel solution

- Several solutions

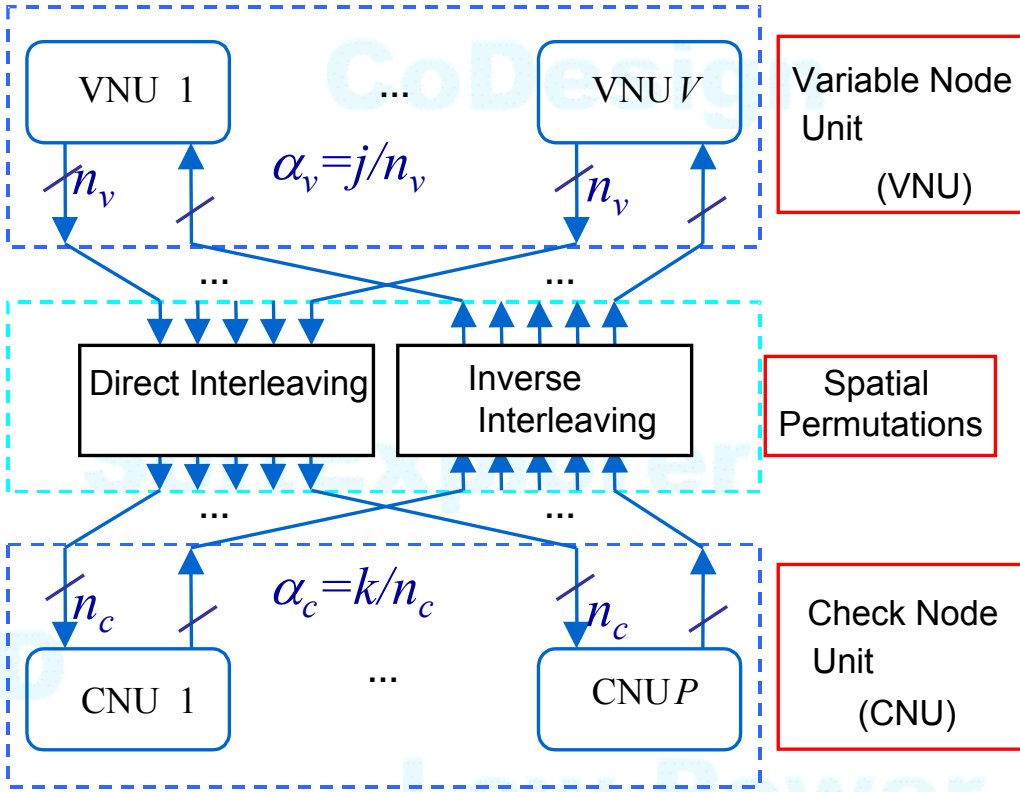




Generic model

- n_v, n_c: number of input/ output ports of the node units

- α_c, α_v : number of clock cycles required to process a CN and a VN

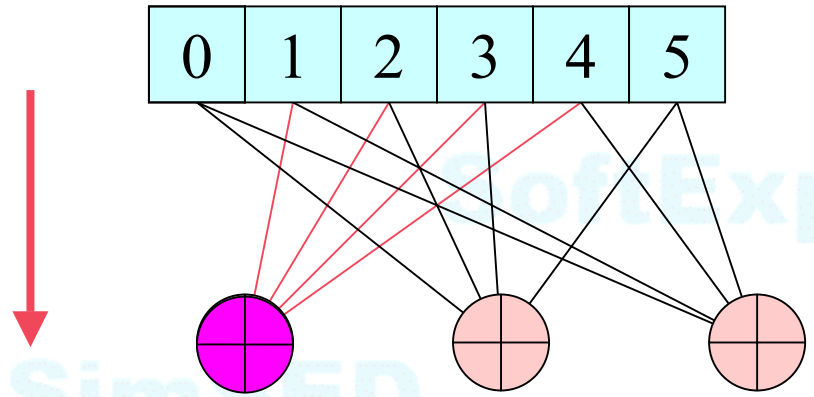


$$P_c = n_c \times P = n_v \times V$$

OUTLINE

- Motivation
- Architecture parameters
  - Parallelism
  - Scheduling
  - Type of generic node (VNU, GNU)
  - Position of the interconnection network
- Example of architecture
- Conclusion

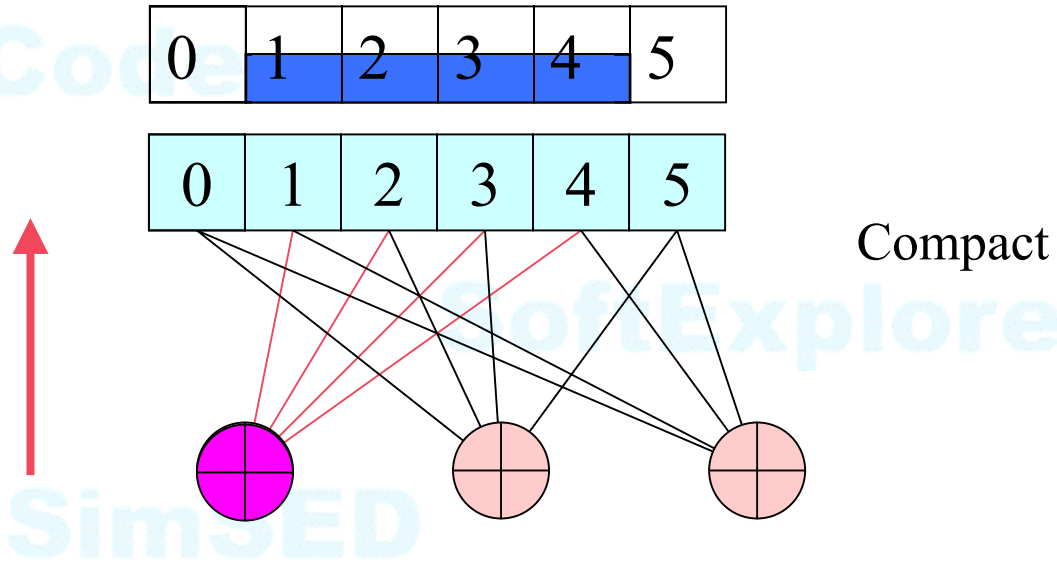
Flooding scheduling



Flooding

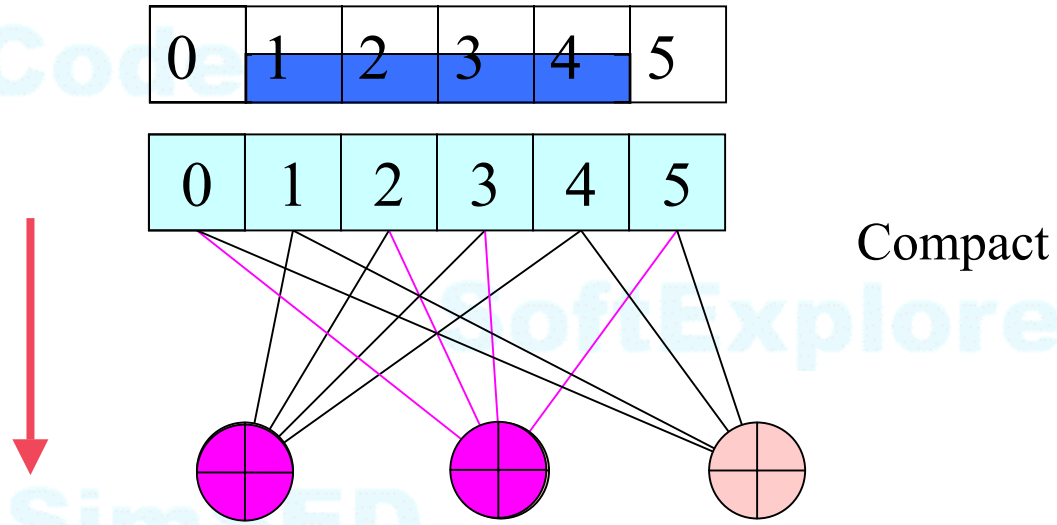
Compact scheduling for the check

Flooding scheduling



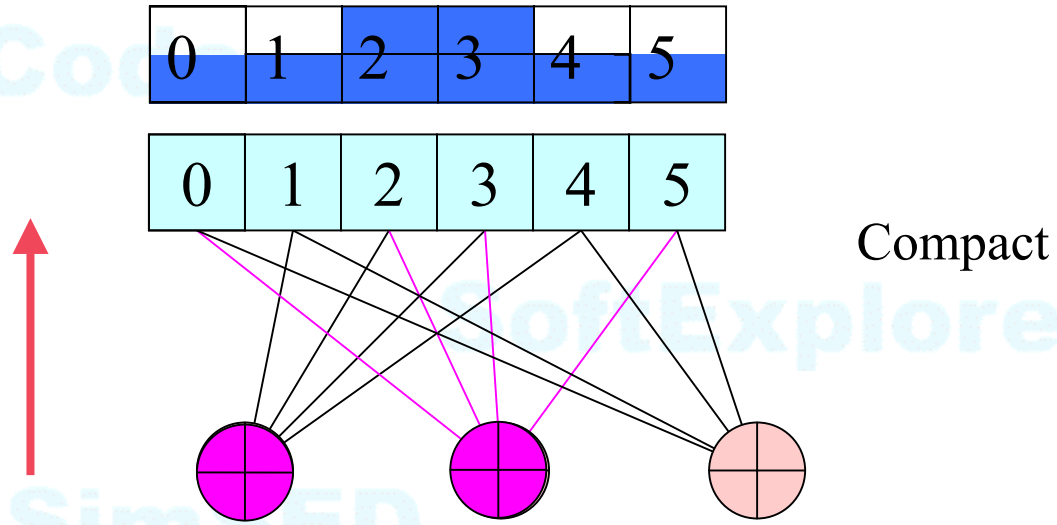
Compact scheduling for the check
Distributed scheduling for the variable, slow update

Flooding scheduling



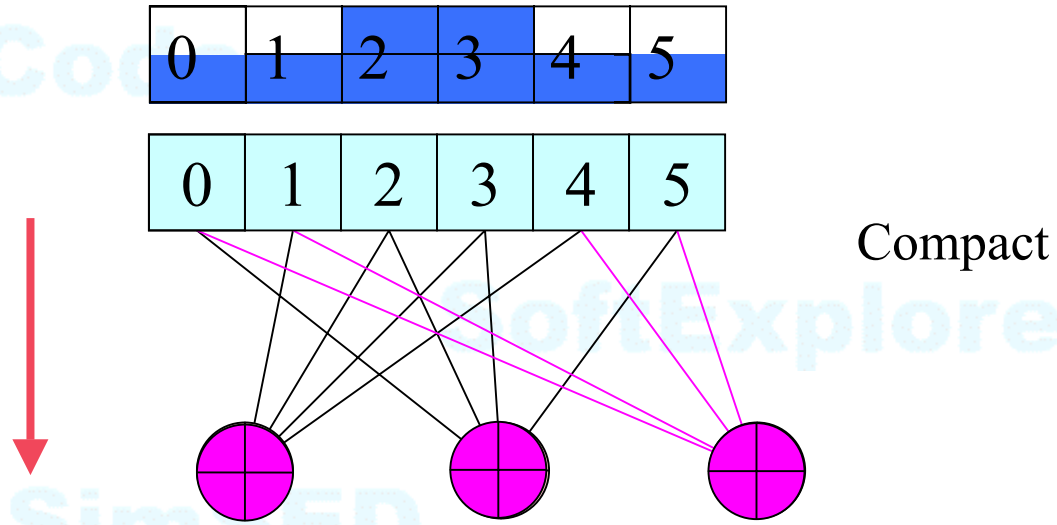
Compact scheduling for the check
Distributed scheduling for the variable, slow update

Flooding scheduling



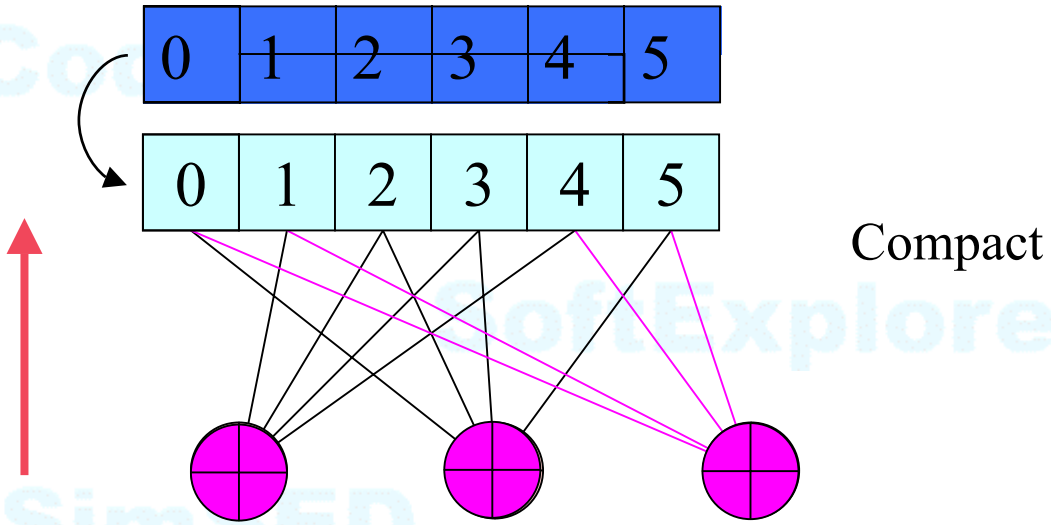
Compact scheduling for the check
Distributed scheduling for the variable, slow update

Flooding scheduling



Compact scheduling for the check
Distributed scheduling for the variable, slow update

Flooding scheduling

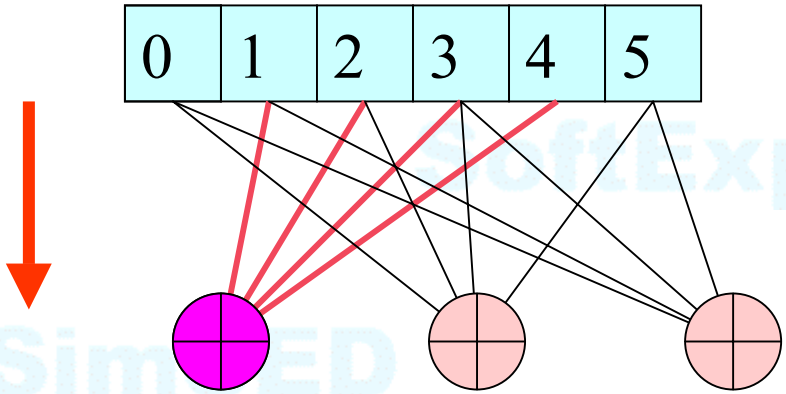


Compact scheduling for the check
Distributed scheduling for the variable, slow update

Horizontal scheduling (turbo-scheduling)

011110
101101
110101

LDPC Matrix

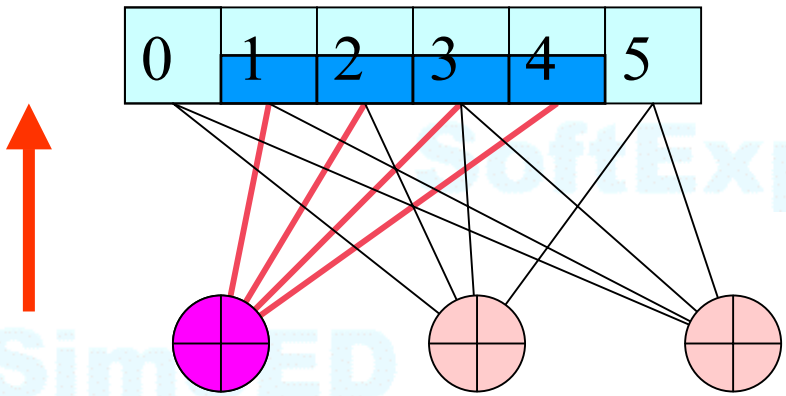


Compact scheduling for the check

Horizontal scheduling (turbo-scheduling)

011110
101101
110101

LDPC Matrix

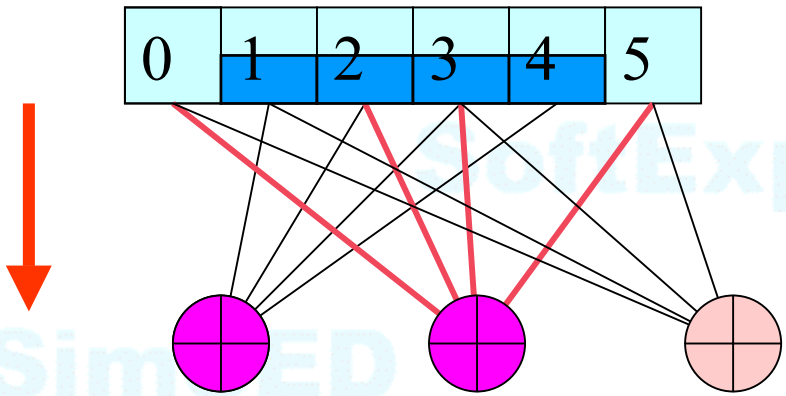


Compact scheduling for the check
Distributed scheduling for the variable, fast update

Horizontal scheduling (turbo-scheduling)

011110
101101
110101

LDPC matrix

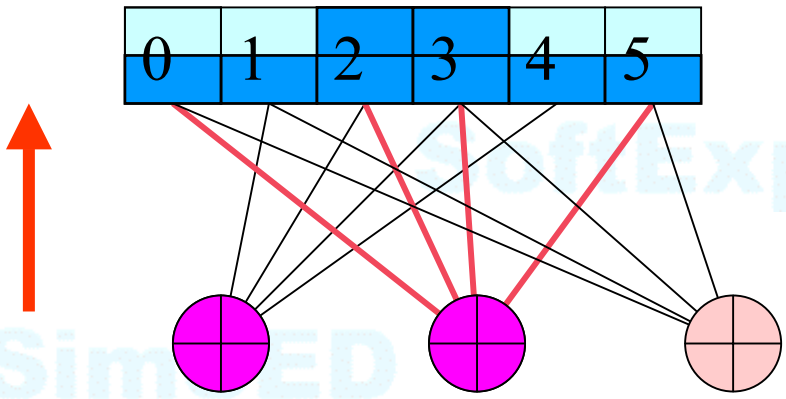


Compact scheduling for the check
Distributed scheduling for the variable, slow update

Horizontal scheduling (turbo-scheduling)

011110
101101
110101

Matrice LDPC



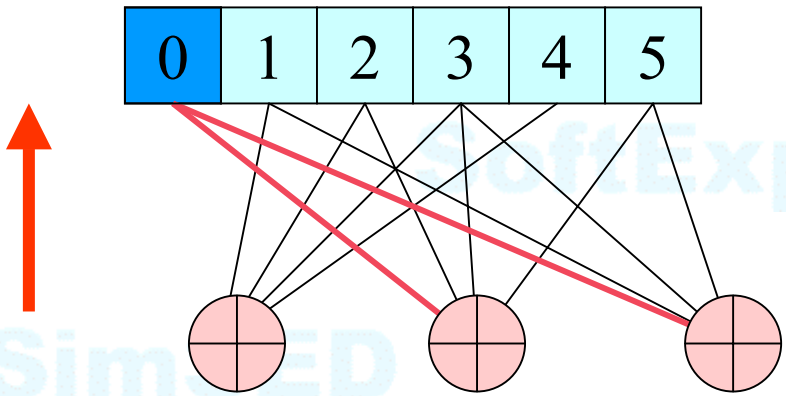
Number of iteration
divided by 1/2

Compact scheduling for the check
Distributed scheduling for the variable, slow update

Vertical scheduling (Fossorier)

011110
101101
110101

LDPC matrix

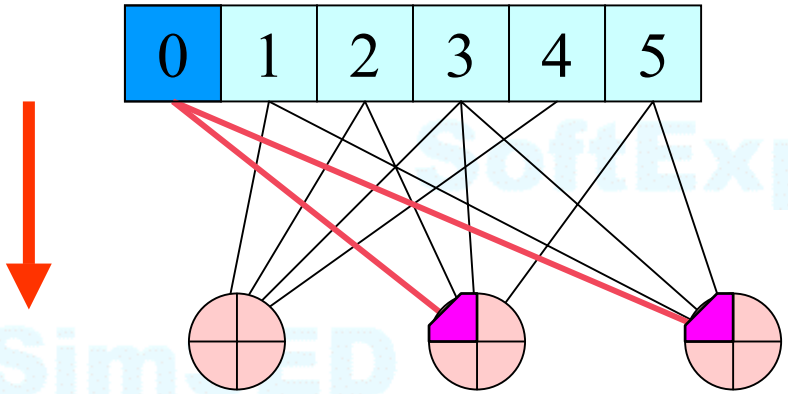


Compact scheduling for the check
Distributed scheduling for the variable, slow update

Vertical scheduling (Fossorier)

011110
101101
110101

LDPC Matrix

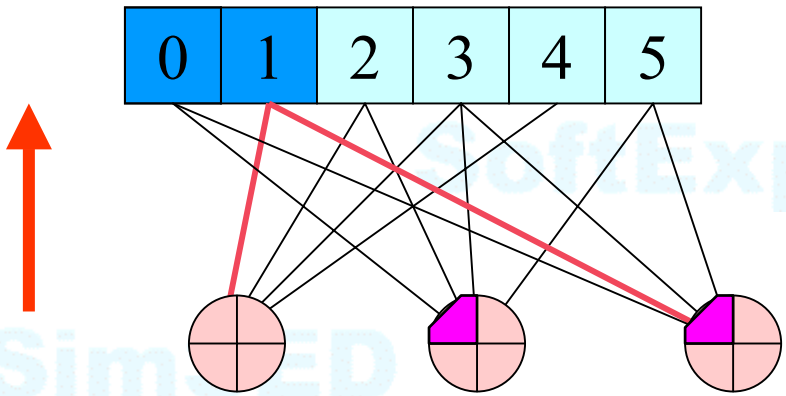


Distributed scheduling for the check, fast update
Compact scheduling for the variable

Vertical scheduling (Fossorier)

011110
101101
110101

LDPC Matrix

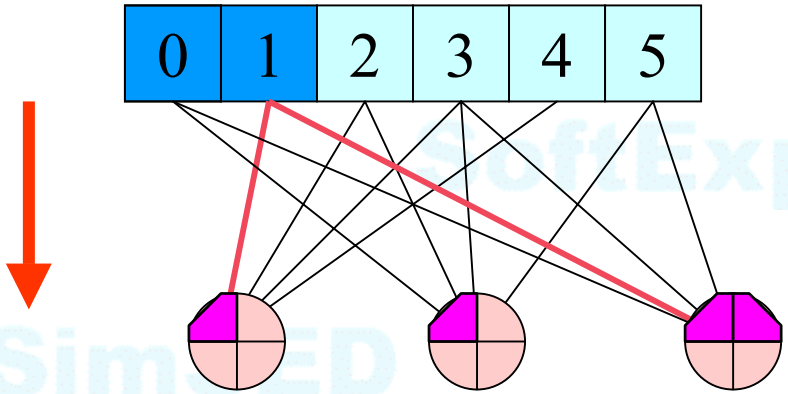


Distributed scheduling for the check, fast update
Compact scheduling for the variable

Vertical scheduling (Fossorier)

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

LDPC matrix



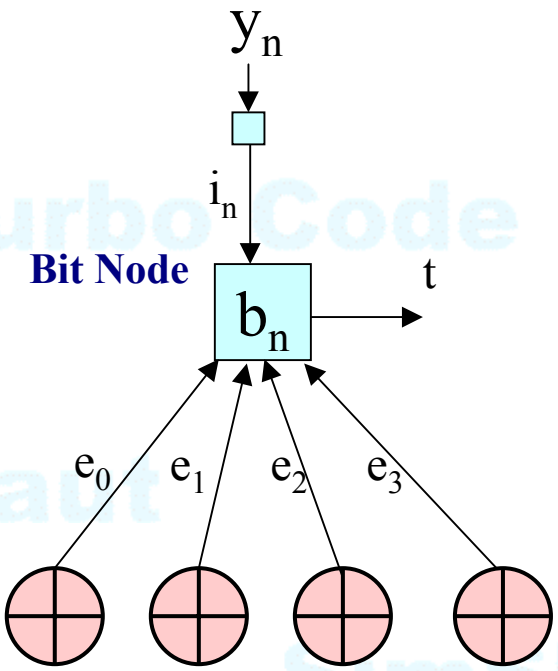
Number of iteration divided by 1/2

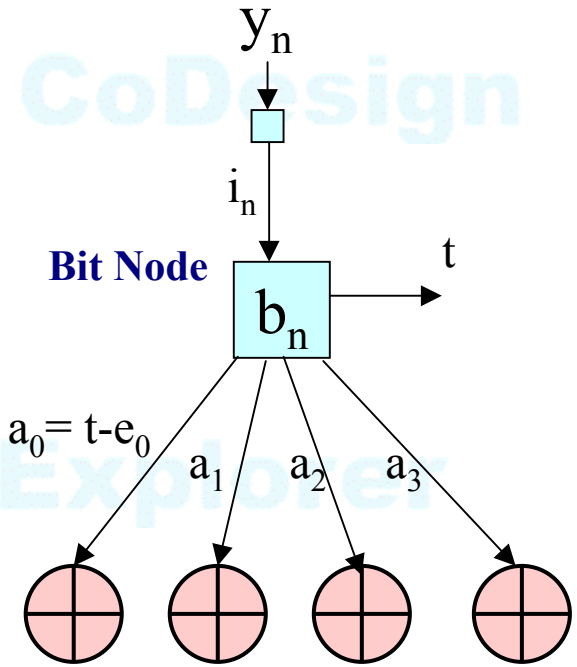
Distributed scheduling for the check, fast update
Compact scheduling for the variable

OUTLINE

- Motivation
- Architecture parameters
  - Parallelism
  - Scheduling
  - Type of generic node (VNU, GNU)
  - Position of the interconnection network
- Example of architecture
- Conclusion

Computation of a variable node

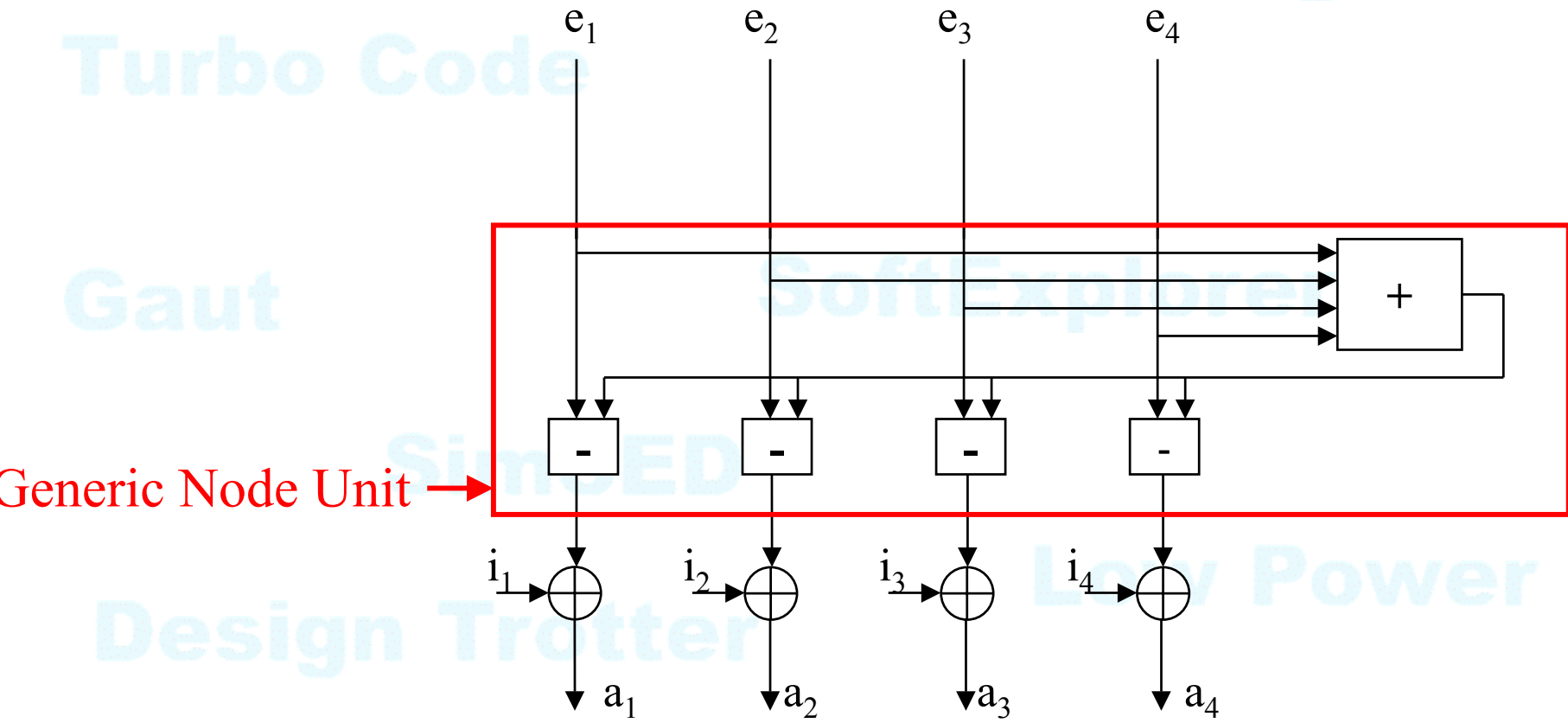




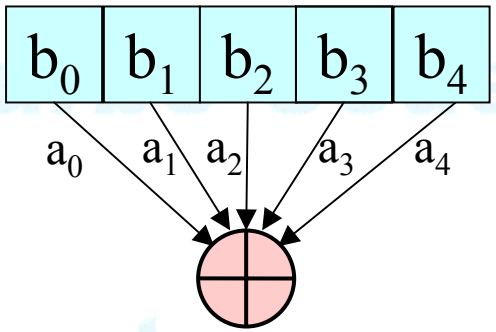
$$t = i_n + \sum_m e_m$$

$$a_i = i_n + \sum_{m,m \neq i} e_i = t - a_i$$

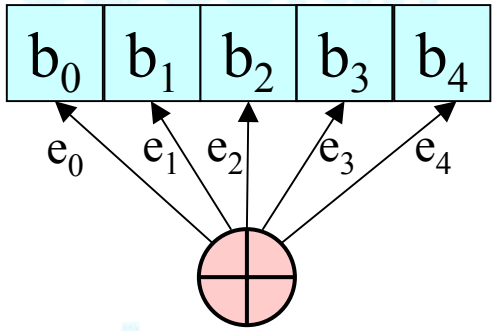
Variable node: data flow graph



Computation of a parity check node



Check Node

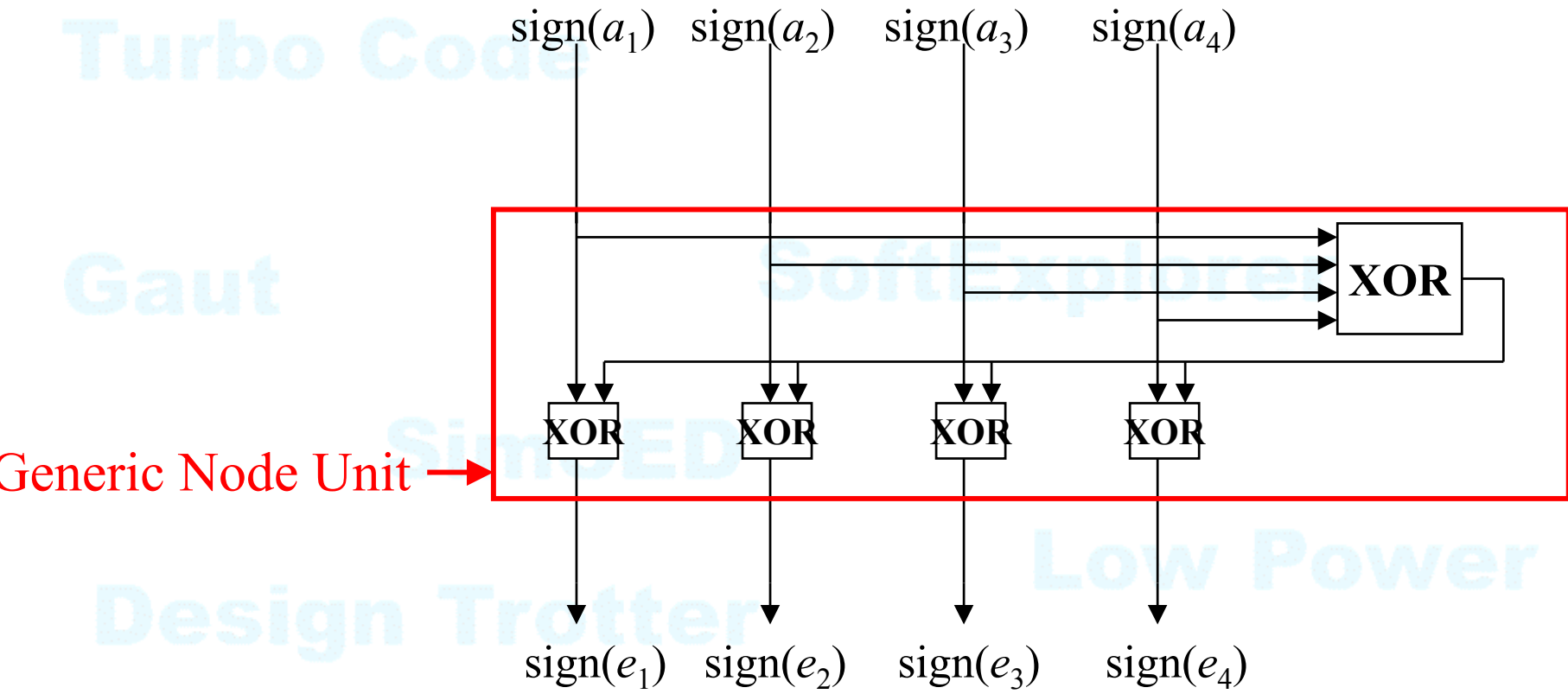


Check Node

$$e_i = \bigoplus_{j \neq i} a_i = a_1 \oplus a_2 \ldots \oplus a_{i-1} \oplus a_{i+1} \ldots \oplus a_{dc}$$

Two methods => exact: probability or frequency domain
=> approximated: min-sum and its variantes

Parity check node in the frequency domain (sign)



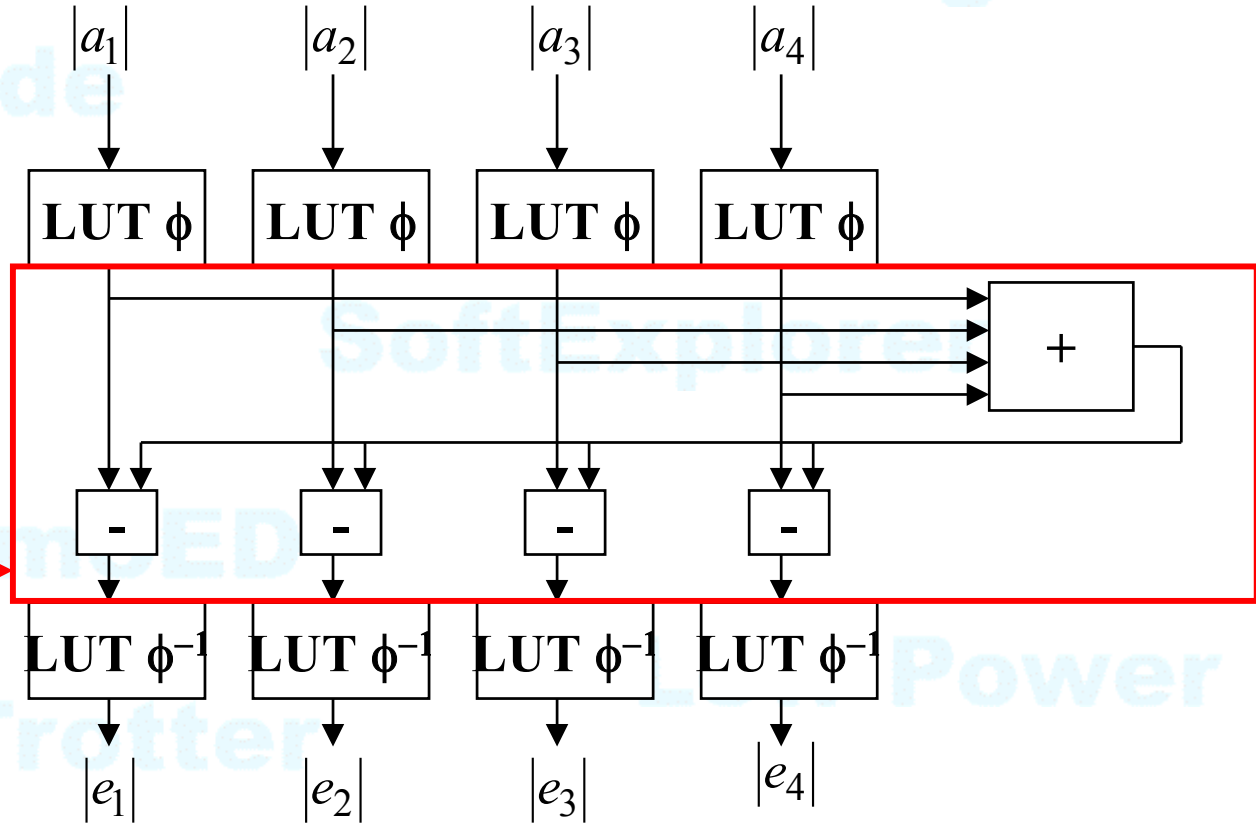
Parity check node in the frequency domain (module)

$$|e_k| = \Phi^{-1}\left(\sum_{i \neq j} \Phi(|a_k|)\right)$$

with

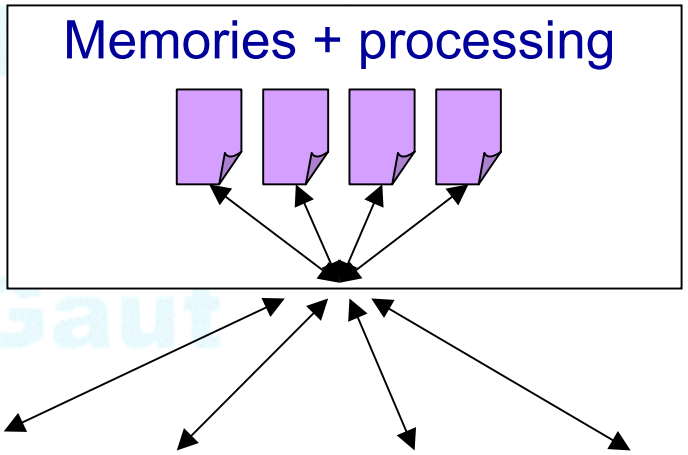
$$\Phi(x) = -\ln(\tanh(x/2))$$

Generic Node Unit →



Generic Node Units (exclude the Φ function)

Generic Node unit

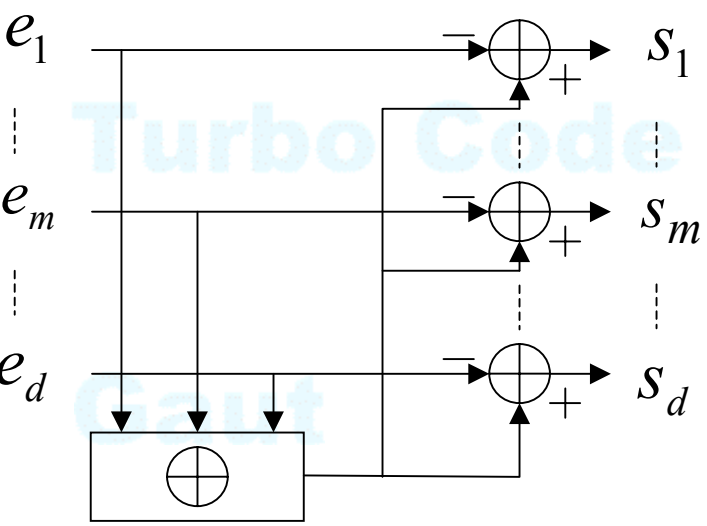


- d inputs e_m and d outputs s_n (d=j or k)
- Input / output law:

$$s_n = \sum_{m \neq n} e_m$$

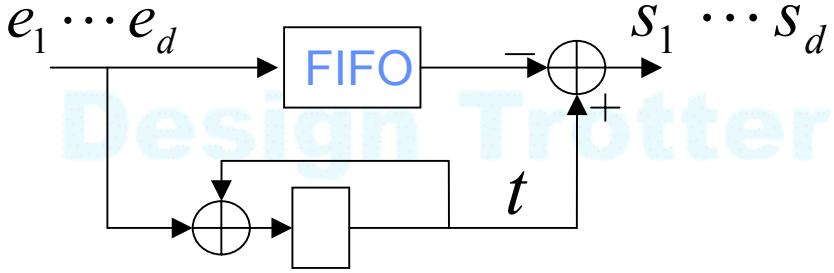
b)

Generic Node
a) Compact mode («no memories »)



Compact mode and parallel implementation

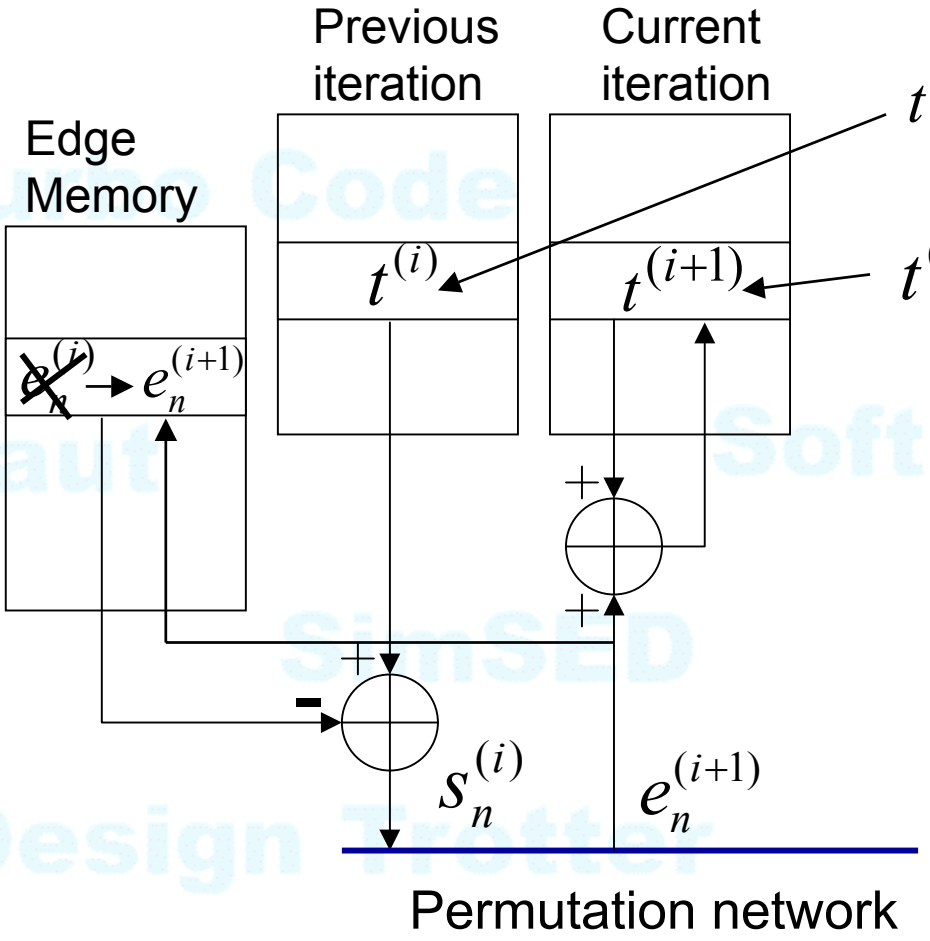
$(\alpha = 1 \rightarrow n=d)$



Compact mode and serial implementation

$(\alpha = d \rightarrow n = 1)$

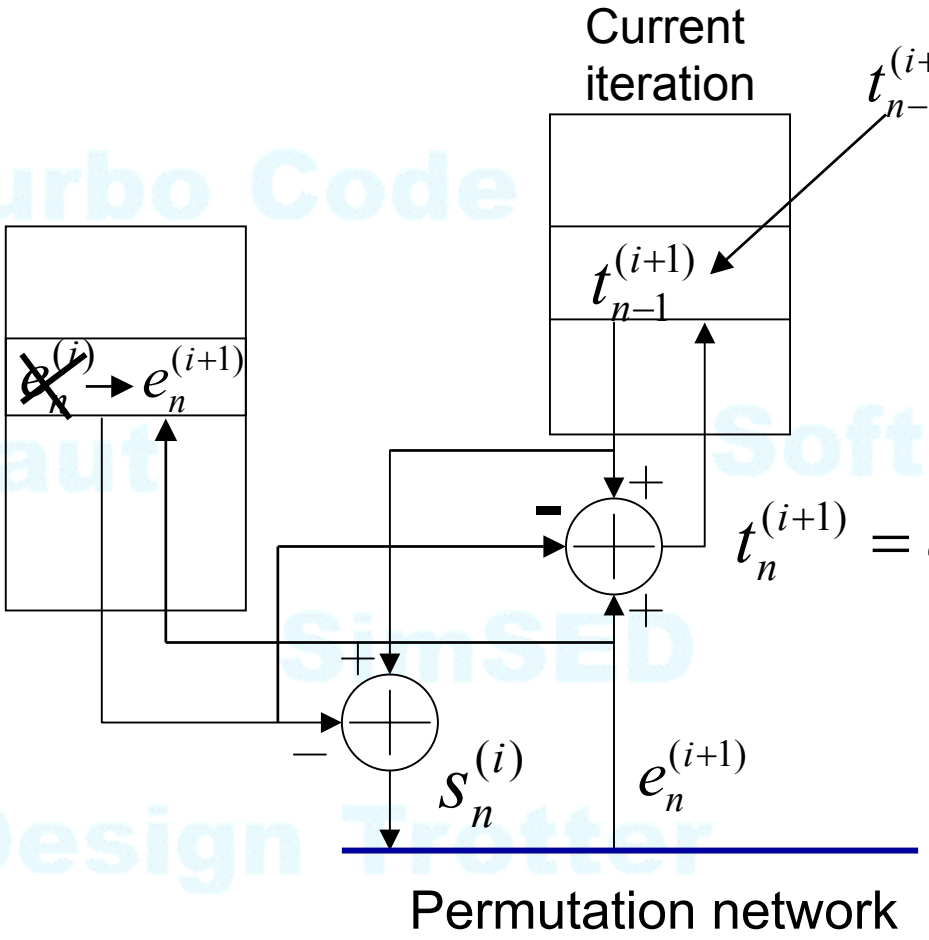
Generic Node
b) Distributed Mode, slow update (Serial)



$$t^{(i)} = e_1^{(i)} + e_2^{(i)} + \ldots + e_d^{(i)}$$

$$t^{(i+1)} = e_1^{(i+1)} + \ldots + e_n^{(i+1)}$$

Generic Node:
c) Distributed Mode, fast update



$$t_{n-1}^{(i+1)} = e_1^{(i+1)} + e_{n-1}^{(i+1)} + e_n^{(i)} + \ldots + e$$

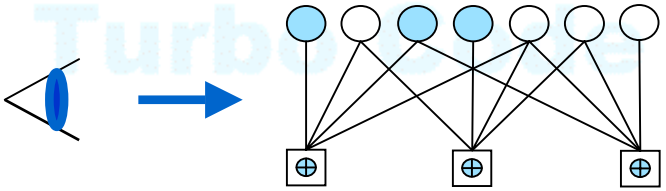
$$t_n^{(i+1)} = e_1^{(i+1)} + e_{n-1}^{(i+1)} + e_n^{(i+1)} + \ldots + e$$

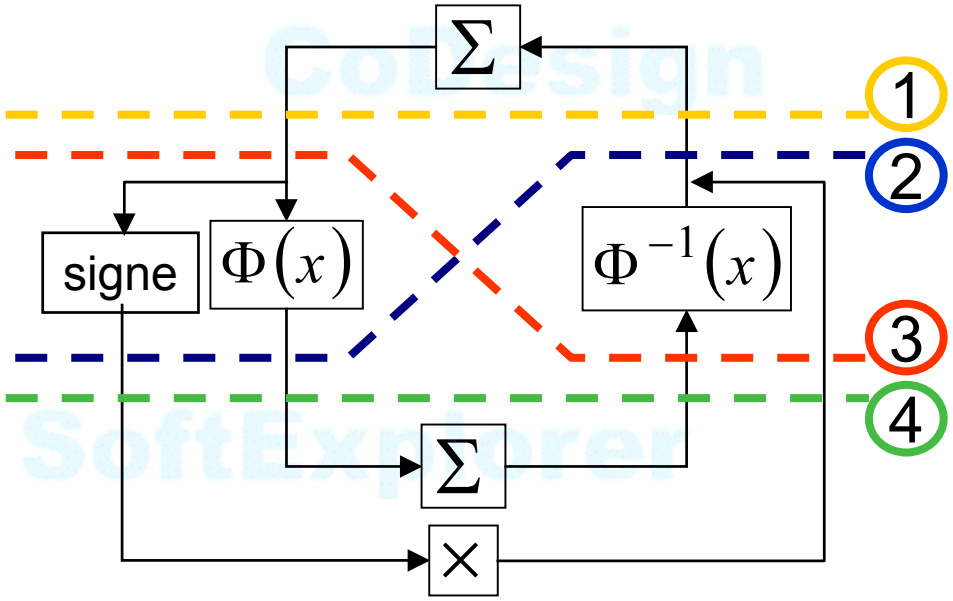
$t^{(i+1)}$ is changing all along the iteration

OUTLINE

- Motivation
- Architecture parameters
  - Parallelism
  - Scheduling
  - Type of generic node (VNU, GNU)
  - Position of the interconnection network
- Example of architecture
- Conclusion

Position of the interconnexion network





or min sum…

Network position		1	2	3	4
Generic Node ⊗	VNU	Σ	$\Phi \circ \Sigma$	$\Sigma \circ \Phi^{-1}$	$\Phi \circ \Sigma \circ \Phi^{-1}$
	CNU	$\Phi^{-1} \circ \Sigma \circ \Phi$	$\Phi^{-1} \circ \Sigma$	$\Sigma \circ \Phi$	Σ
		× (sign product)			

Result : Generic description of the decoder architecture

- Node Unit Architecture
- Parallelism : $P, V, \alpha_{CNU}, a_{VNU}$
- Processeur control (compact / distributed)
- Shuffle network position (1,2,3,4)

Many combinations of the parameters are possible:

Control mode combinations

CNU			VNU: Compact	VNU: Distributed, Slow Update	VNU: Distributed, Fast Update
Compact			Parallel Flooding	Flooding along CNU	Horizontal Shuffle
Distributed	Slow Update		Flooding along VNU	Edge by edge control	
	Fast Update		Vertical Shuffle		

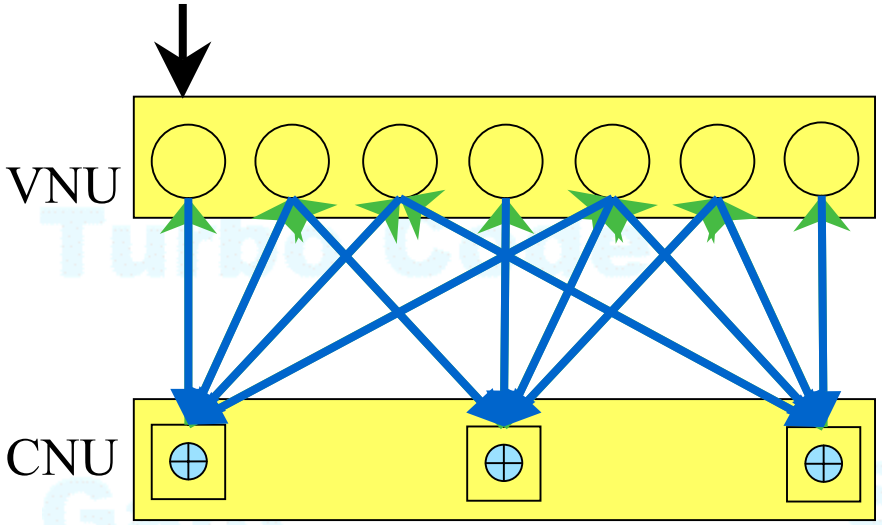
OUTLINE

- Motivation
- Architecture parameters
- Example of architecture
- Conclusion

Vertical Shuffle ad-hoc architecture (1)

- Originally proposed as Shuffle-BP (Zhang & Fossorier, 2002)
- Combination of:
  - Compact Mode for VNU
  - Distributed Mode with fast update for CNU

Vertical Shuffle ad-hoc architecture (1)



$$E_{m,n} = I_n + \sum_{m' \in M(n) \setminus m} T_{m',n}$$

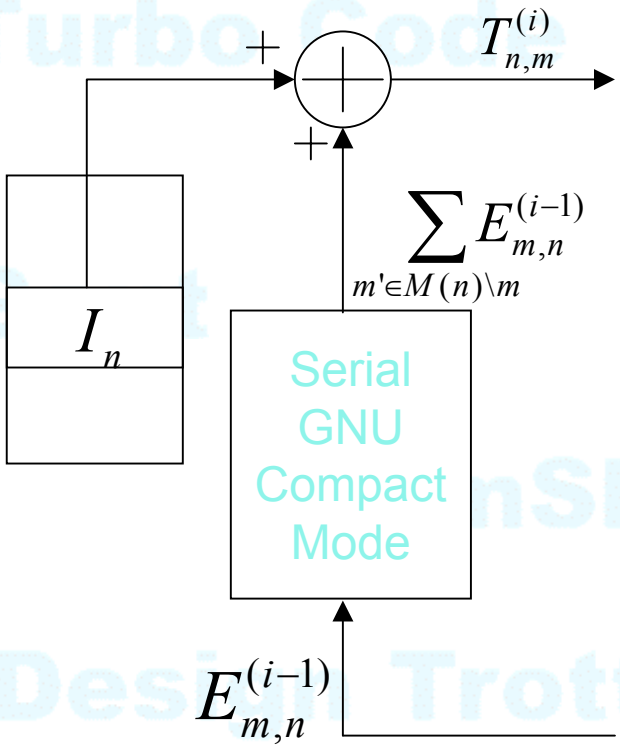
$$T_{m,n} = \Phi^{-1}\left(\sum_{n' \in N(m) \setminus n} \Phi(E_{m,n'}) \right)$$

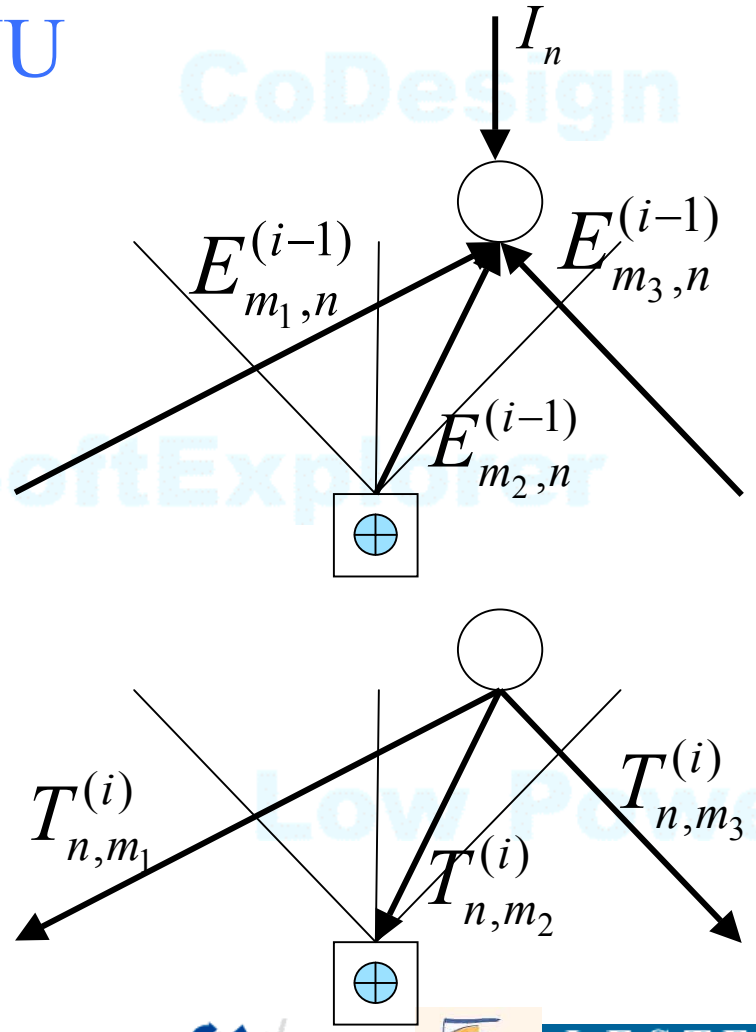
$$\Phi(x) = -\ln\left(\tanh\left(x/2\right)\right)$$

$$Q_{m,n} = \Phi(E_{m,n'}) \qquad R_m = \sum_{n \in N(m)} \Phi(E_{m,n}) \qquad T_{m,n} = \Phi^{-1}\left(R_n - Q_{m,n}\right)$$

Vertical Shuffle ad-hoc architecture (2)

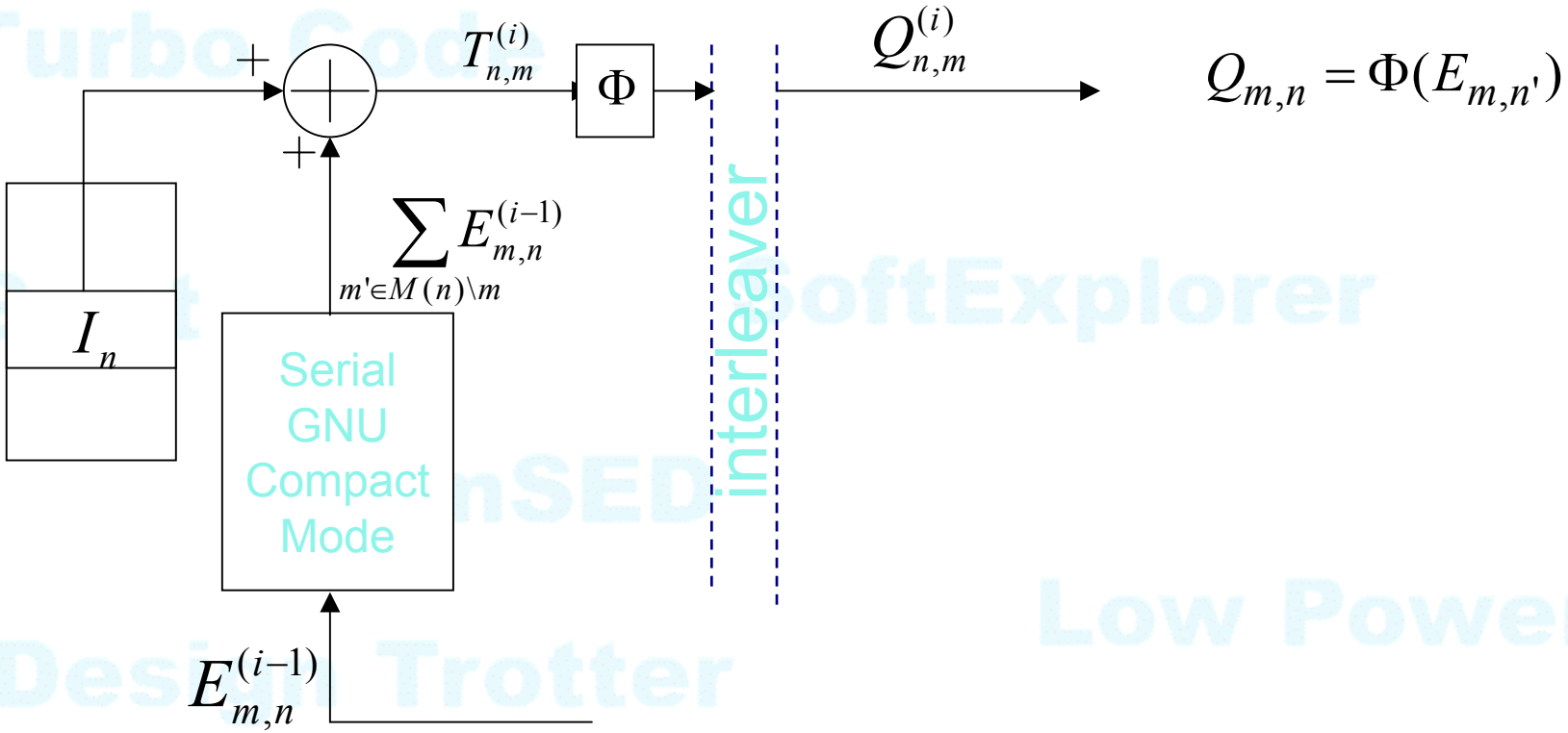
- Compact, Serial VNU





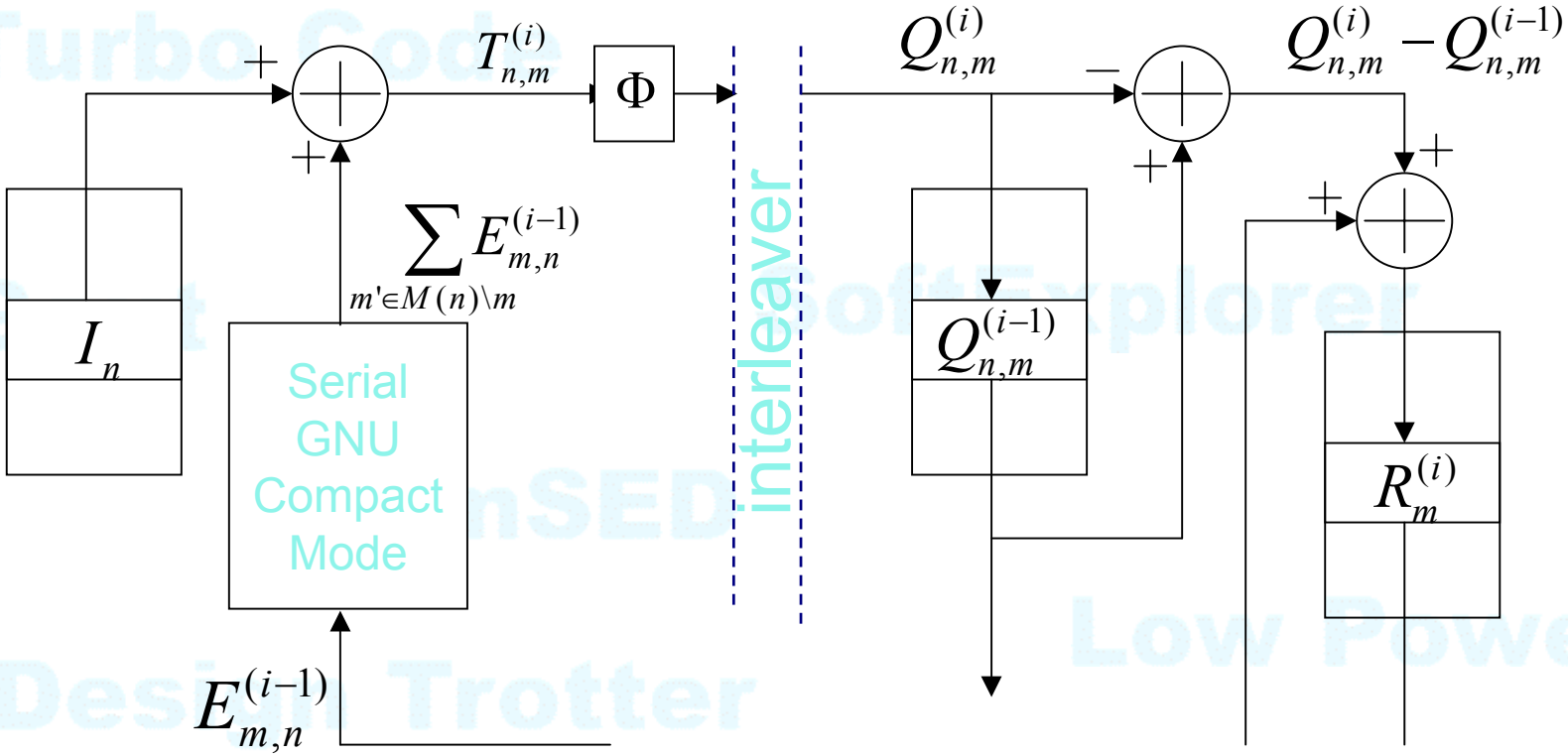
Vertical Shuffle ad-hoc architecture (3)

- Go through the ϕ functions and interleaver



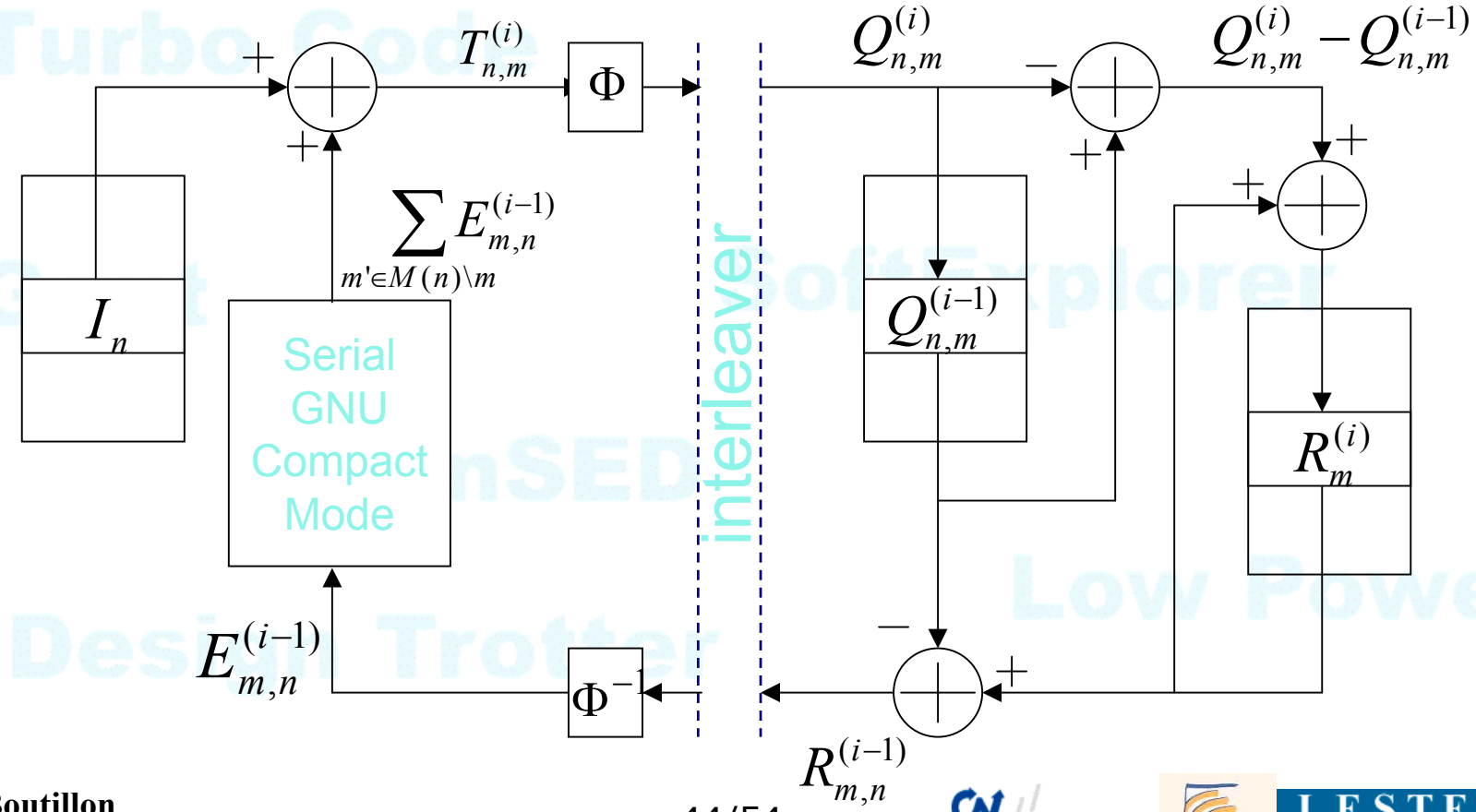
Vertical Shuffle ad-hoc architecture(4)

- Save $Q_{n,m}$ and update R_m (fast update)



Vertical Shuffle ad-hoc architecture (5)

$$R_m = \sum_{n \in N(m)} \Phi(E_{m,n})$$



OUTLINE

- Motivation
- Architecture parameters
- Example of architecture
- Conclusion

Conclusion

Proposition of partial framework for description, analysis and synthesis of LDPC decoder architectures.

Based on the formalism, proposition of an architecture for vertical scheduling

Extend methodology to all LDPC architecture (to be done)

IP MIMO CoDesign Turbo Code Gaut SoftExplorer SimSED Low Power Design Trotter

Thank you for your attention !