Achieving degradation tolerance in a hardware accelerator with parallel functional units

Tomohiro Yoneda
National Institute of Informatics

Masashi Imai
Univ. of Tokyo

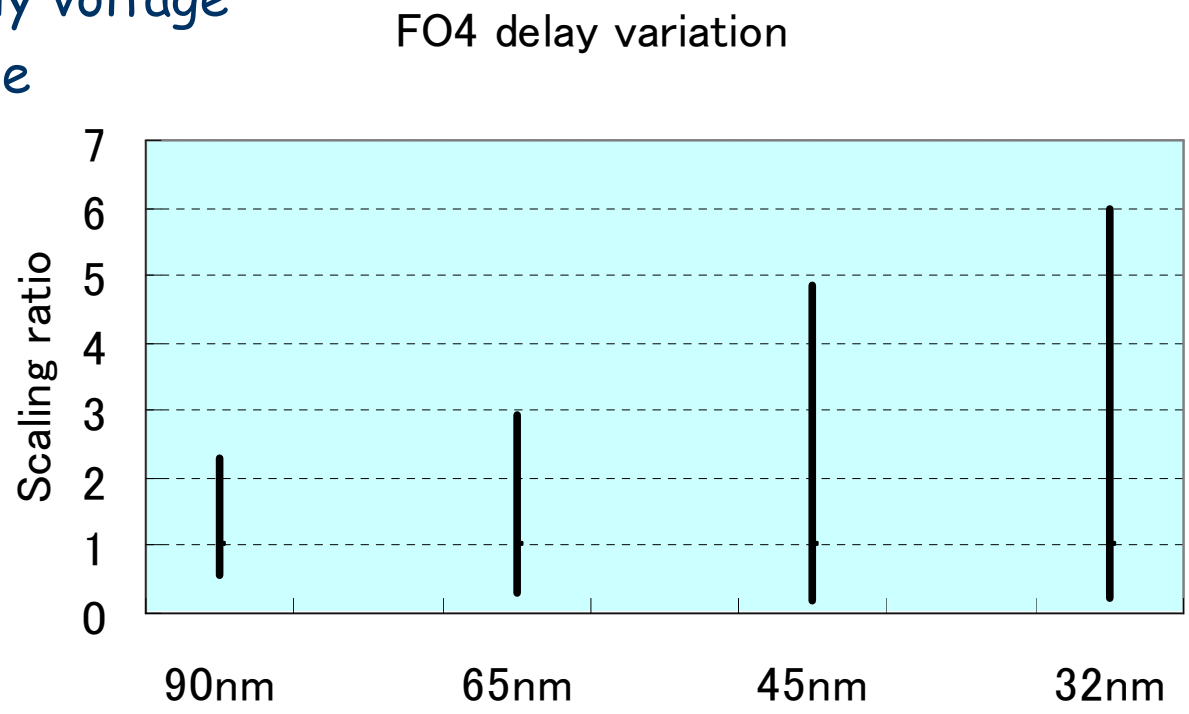
Atsushi Matsumoto
Tohoku Univ.

Hiroshi Saito
Univ. of Aizu

National Institute of Informatics
国立情報学研究所

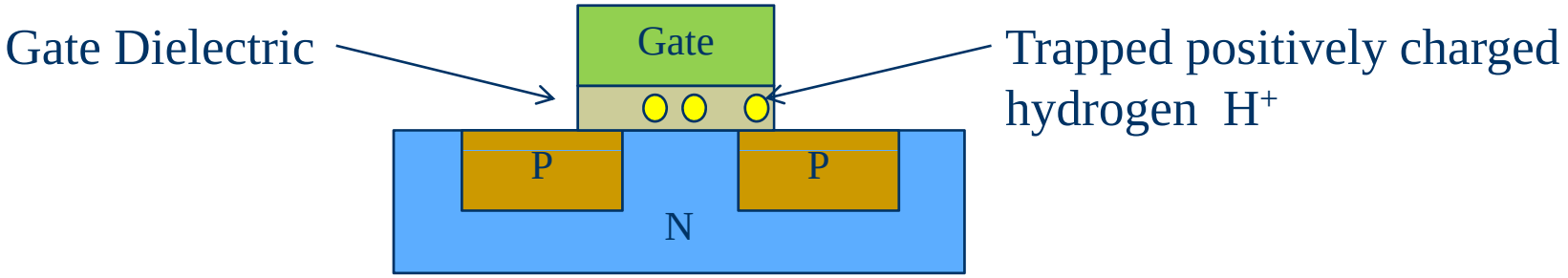
Background (1)

- Advances in semiconductor technologies
  - Larger variations in transistor performance
    - process
    - power supply voltage
    - temperature



Background (2)

- Advances in semiconductor technologies
  - New types of faults
    - NBTI (Negative Bias Temperature Instability)
      - occurs in PMOS Tr. stressed with negative gate voltages at elevated temperatures



    - HCI (Hot Carrier Injection)
    - Threshold voltage V_T increased, absolute drain current I_{Dsat} decreased $\Rightarrow$ increased delay

Goal

- Degradation
  - statically caused
    - slow transistors due to process variation
  - dynamically caused
    - slowdowns
      - in low voltage, high temperature (local) area
      - by NBTI, HCI, etc
- Goal
  - obtain
    - correct functionality
    - almost non-degraded performance

  autonomously with no spare or monitor units

Approach

- Use asynchronous circuit implementation
  - Simple synchronous circuit implementation causes sudden incorrect operation unless large timing margins are given
  - Asynchronous circuits can always guarantee correct operation without unnecessary margins
  - ✖ Still possible due to performance degradation that producing final results is delayed and deadline is missed
- Send less data to degraded functional units by completion signal based data-flow control

Targeted application

- Assumption
  - Several or many functional units that can cover each other are available
    - No spare units are needed
    - Used for performance improvement
  - Many parallel computation algorithms
    - SIMD style computation
    - Running example in this work: linear equation solver based on Gauss-Seidel method

Gauss-Seidel Method

- System of linear equations

$$\begin{bmatrix} & & & & \\ a_{i0} & \cdots & a_{ii} & \cdots & a_{i(n-1)} \\ & & & & \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_i \\ \vdots \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} b_0 \\ \vdots \\ b_i \\ \vdots \\ b_{n-1} \end{bmatrix}$$

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right)$$

Gauss-Seidel Method

Step 1: $$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right)$$

$$\begin{bmatrix} x_0^1 \\ \\ \\ \end{bmatrix} = \begin{bmatrix} \frac{1}{a_{00}} & \frac{-a_{01}}{a_{00}} & \frac{-a_{02}}{a_{00}} & \cdots & \frac{-a_{0(n-1)}}{a_{00}} \end{bmatrix} \begin{bmatrix} b_0 \\ x_1^0 \\ x_2^0 \\ \vdots \\ x_{(n-1)}^0 \end{bmatrix}$$

Gauss-Seidel Method

Step 2:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right)$$

$$\begin{bmatrix} x_0^1 \\ x_1^1 \\ \\ \end{bmatrix} = \begin{bmatrix} \frac{-a_{10}}{a_{11}} & \frac{1}{a_{11}} & \frac{-a_{12}}{a_{11}} & \cdots & \frac{-a_{1(n-1)}}{a_{11}} \end{bmatrix} \begin{bmatrix} x_0^1 \\ b_0 \\ x_2^0 \\ \vdots \\ x_{(n-1)}^0 \end{bmatrix}$$

Gauss-Seidel Method

Step 3:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right)$$

$$\begin{bmatrix} x_0^1 \\ x_1^1 \\ x_2^1 \end{bmatrix} = \begin{bmatrix} \frac{-a_{20}}{a_{22}} & \frac{-a_{21}}{a_{22}} & \frac{1}{a_{22}} & \cdots & \frac{-a_{2(n-1)}}{a_{22}} \end{bmatrix} \begin{bmatrix} x_0^1 \\ x_1^1 \\ b_0 \\ \vdots \\ x_{(n-1)}^0 \end{bmatrix}$$

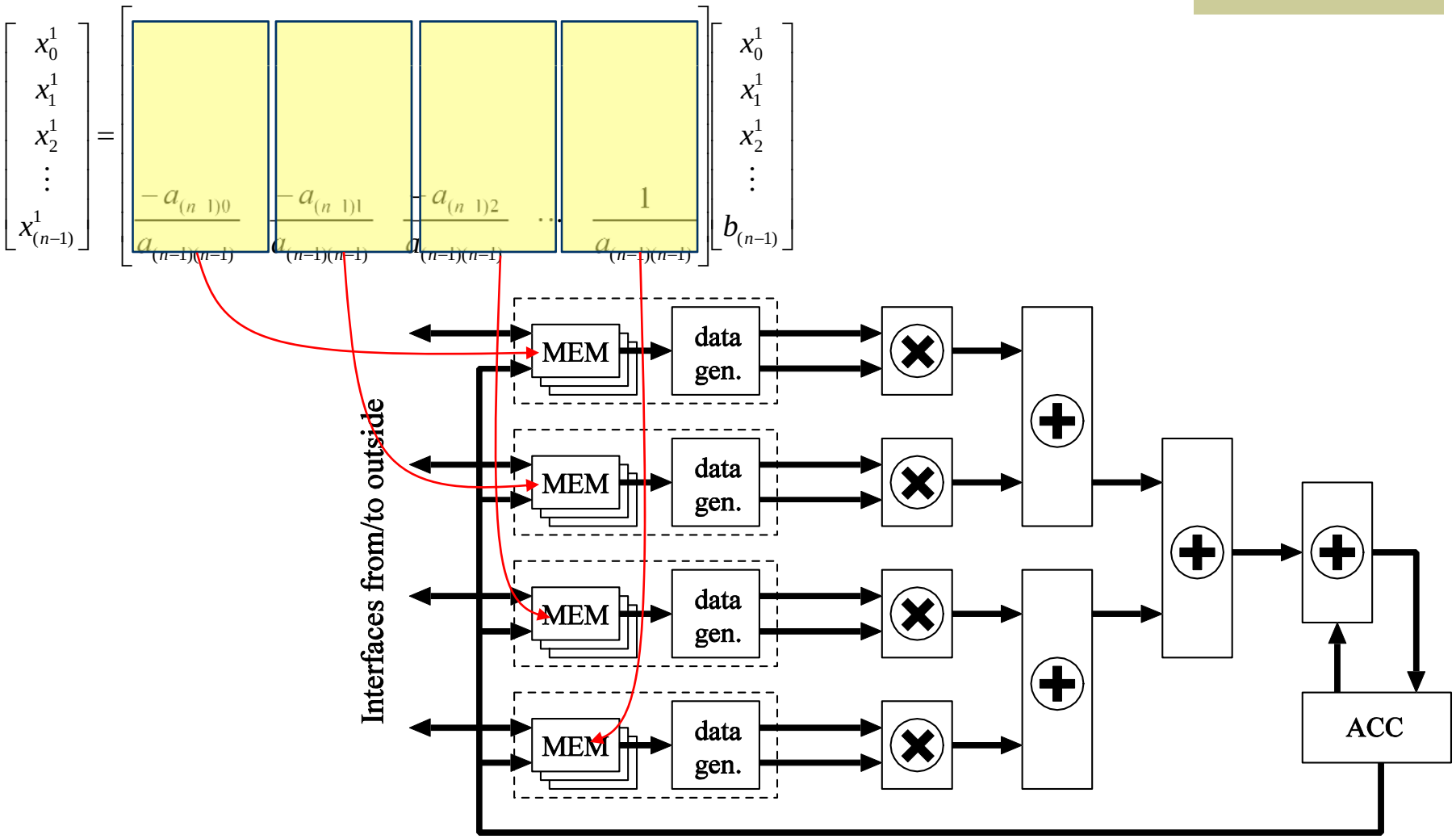
Gauss-Seidel Method

Step *n*:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=0}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}\right)$$

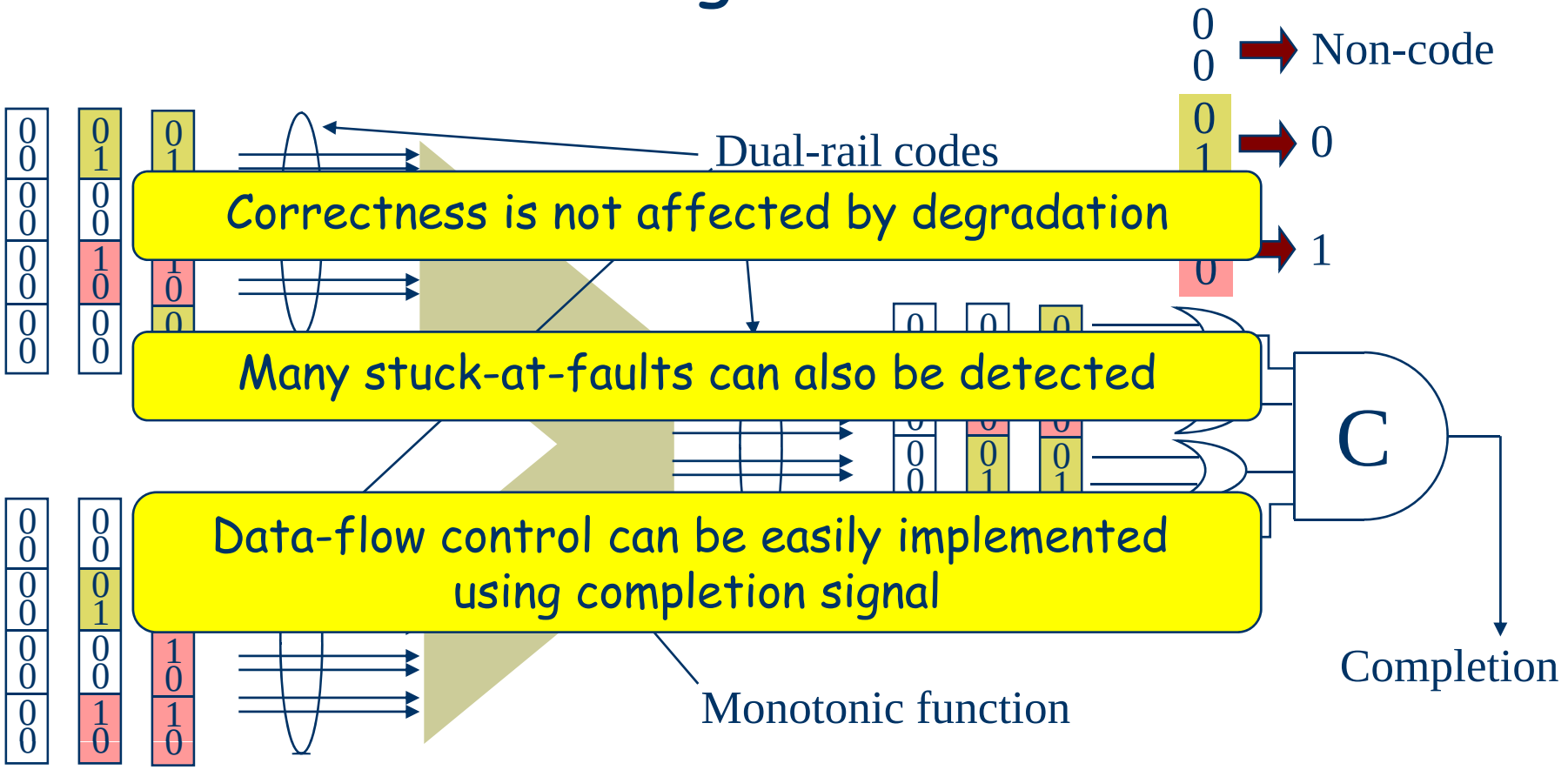
$$\begin{bmatrix} x_0^1 \\ x_1^1 \\ x_2^1 \\ \vdots \\ x_{(n-1)}^1 \end{bmatrix} = \begin{bmatrix} \frac{-a_{(n-1)0}}{a_{(n-1)(n-1)}} & \frac{-a_{(n-1)1}}{a_{(n-1)(n-1)}} & \frac{-a_{(n-1)2}}{a_{(n-1)(n-1)}} & \cdots & \frac{1}{a_{(n-1)(n-1)}} \end{bmatrix} \begin{bmatrix} x_0^1 \\ x_1^1 \\ x_2^1 \\ \vdots \\ b_{(n-1)} \end{bmatrix}$$

Implementation

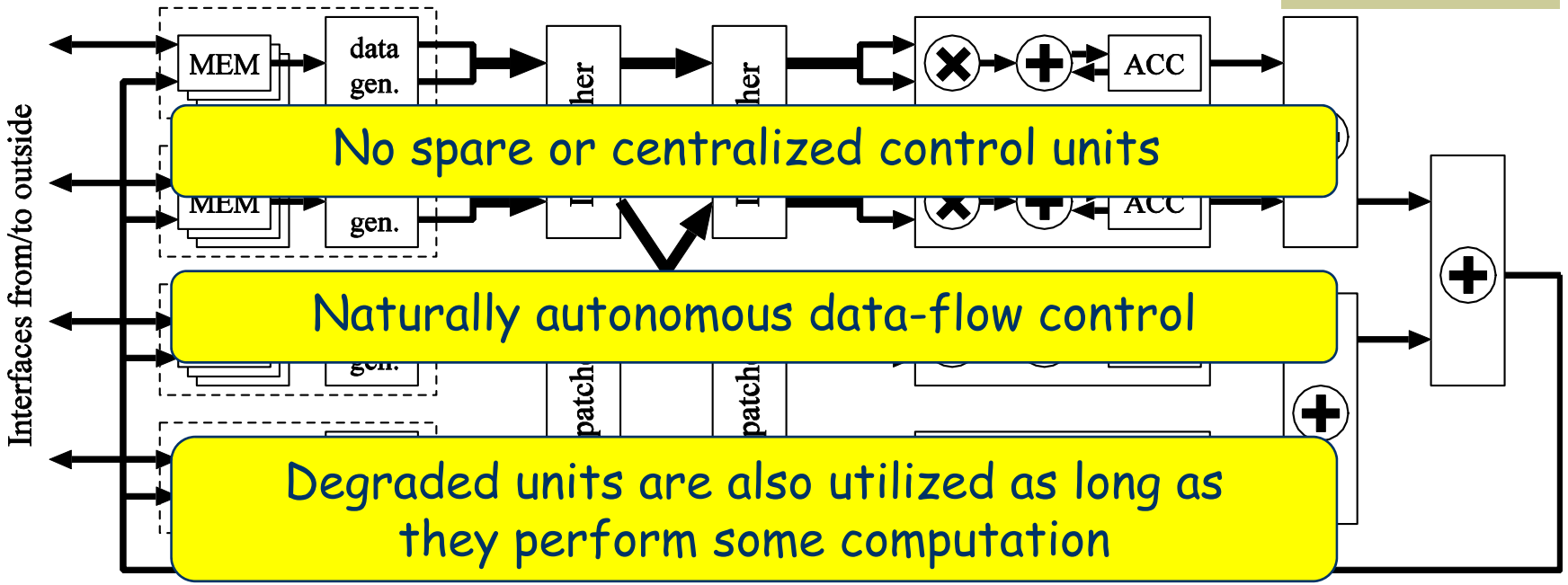


Asynchronous circuits

- Dual-rail encoding method



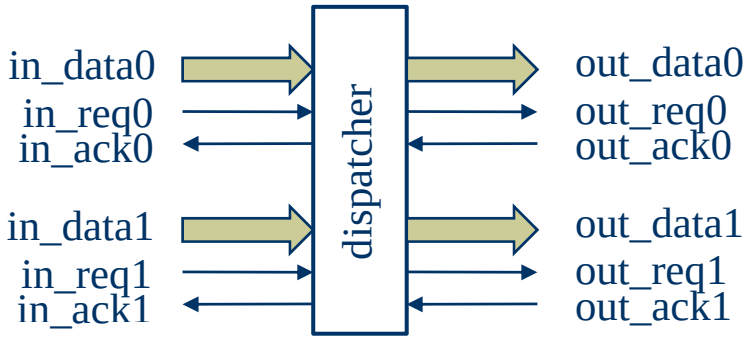
Proposed architecture



- Dispatcher
  - Sends less data to degraded functional units
    - completion signals indicate busy/idle information
    - data can be sent to an idle direction
  - Dispatchers also give completion (Ack) signals to previous stages

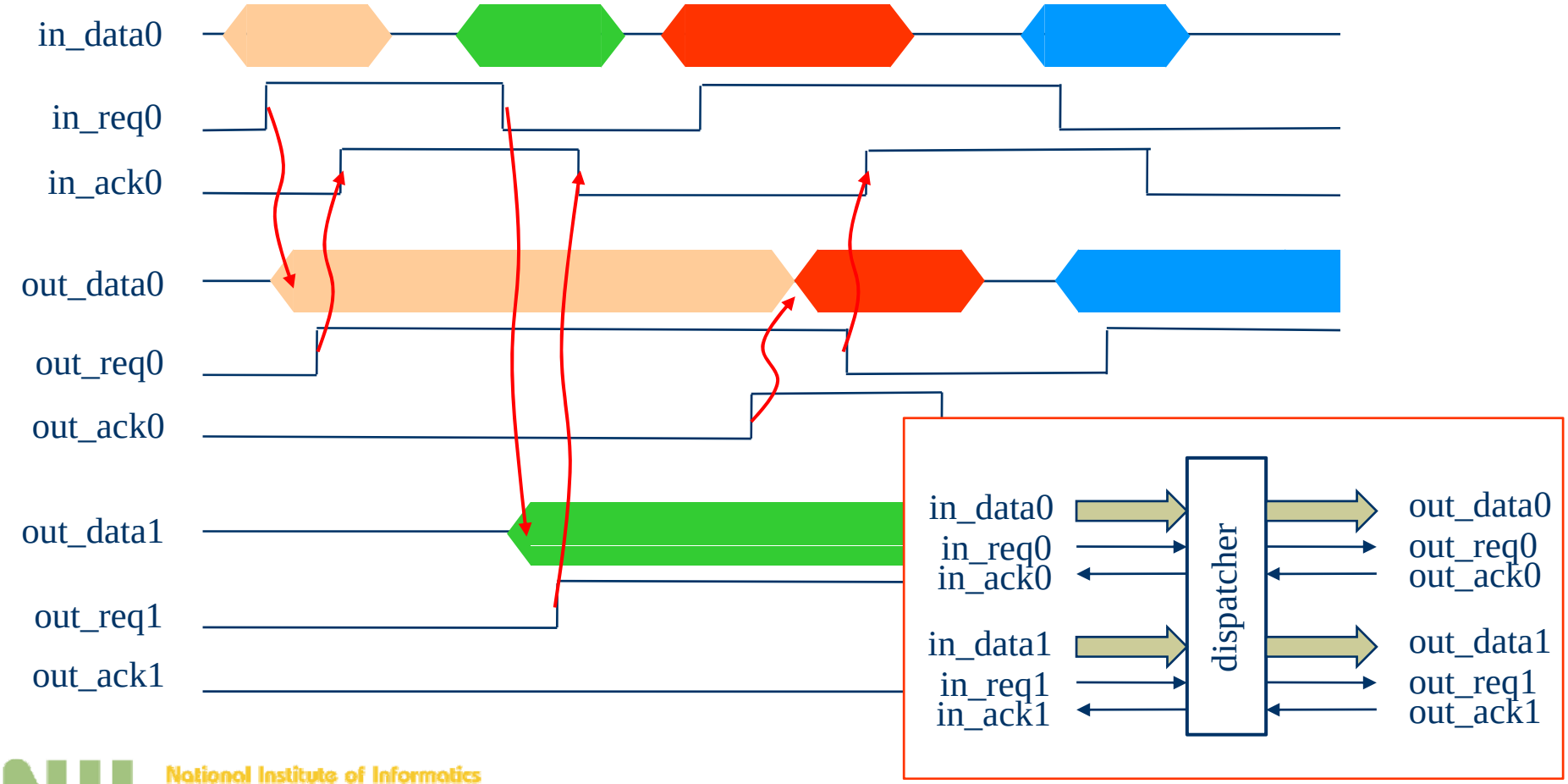
Dispatcher design (1)

- Bundled data method
  - Explicit request signals are used, instead of encoded data
    - Delay elements are used to decide timings
  - Simple, quick, and suitable for non-data manipulating circuit blocks



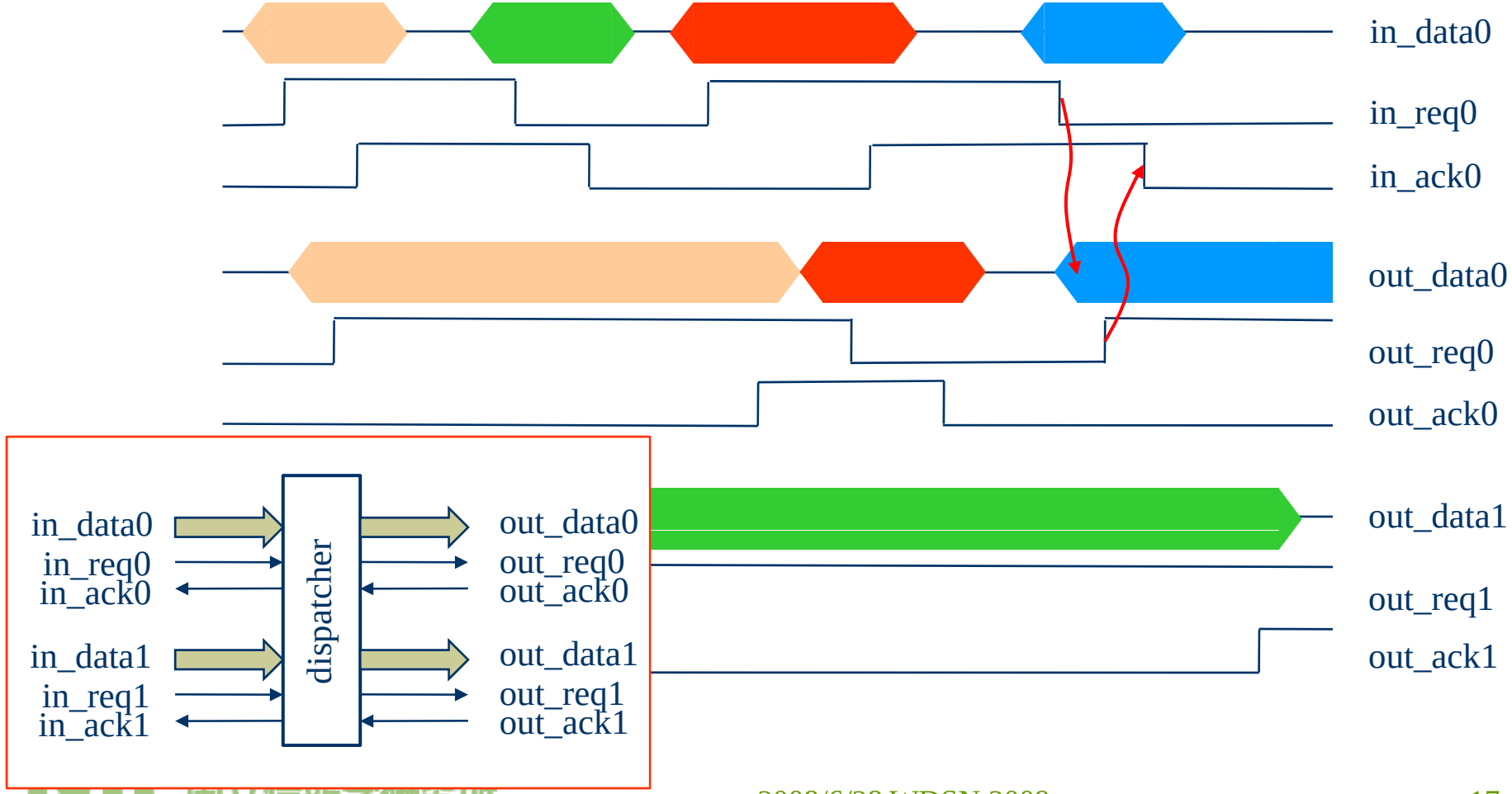
Dispatcher design (2)

- Two phase signaling protocol

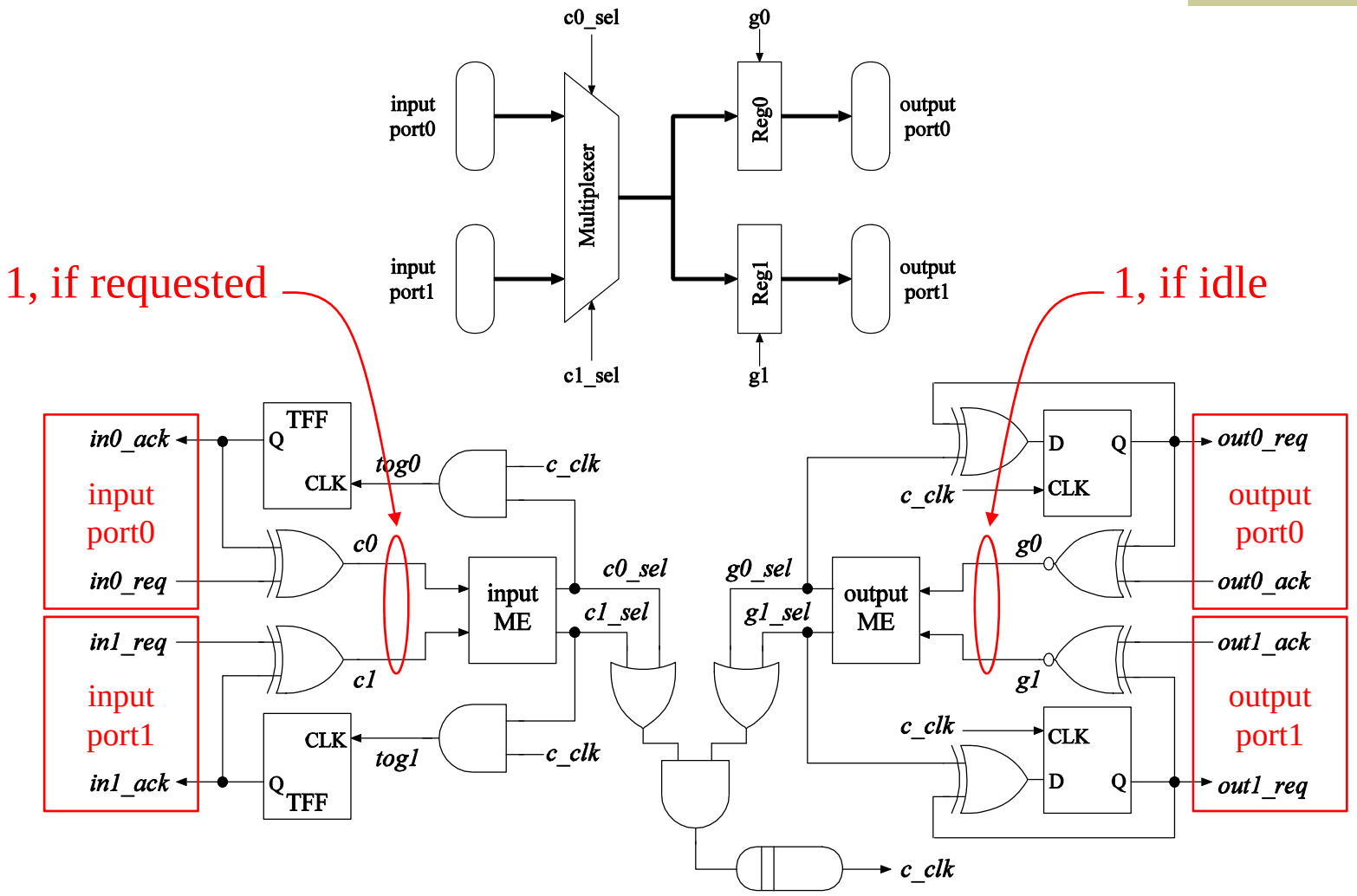


Dispatcher design (2)

- Two phase signaling protocol

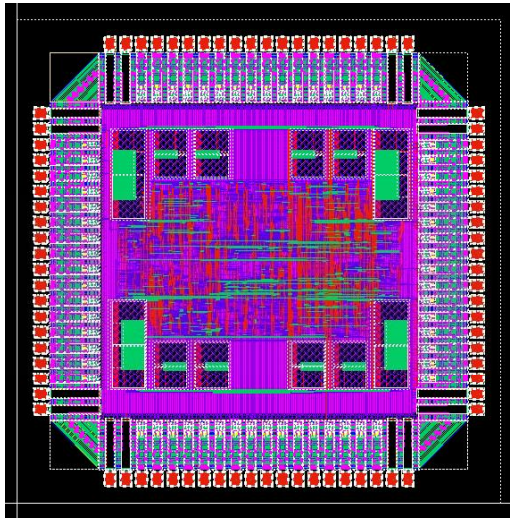


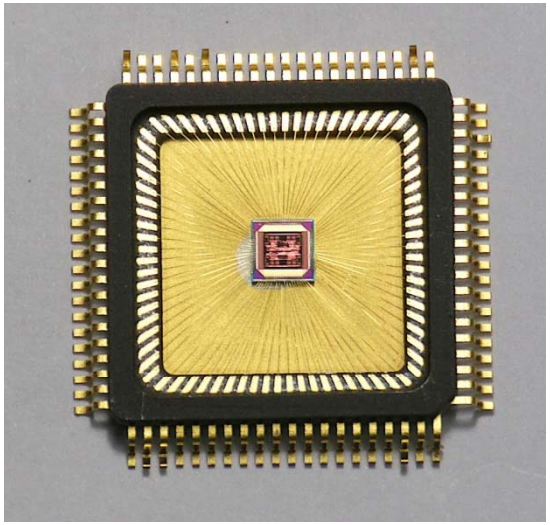
Dispatcher design (3)



Experiments (1)

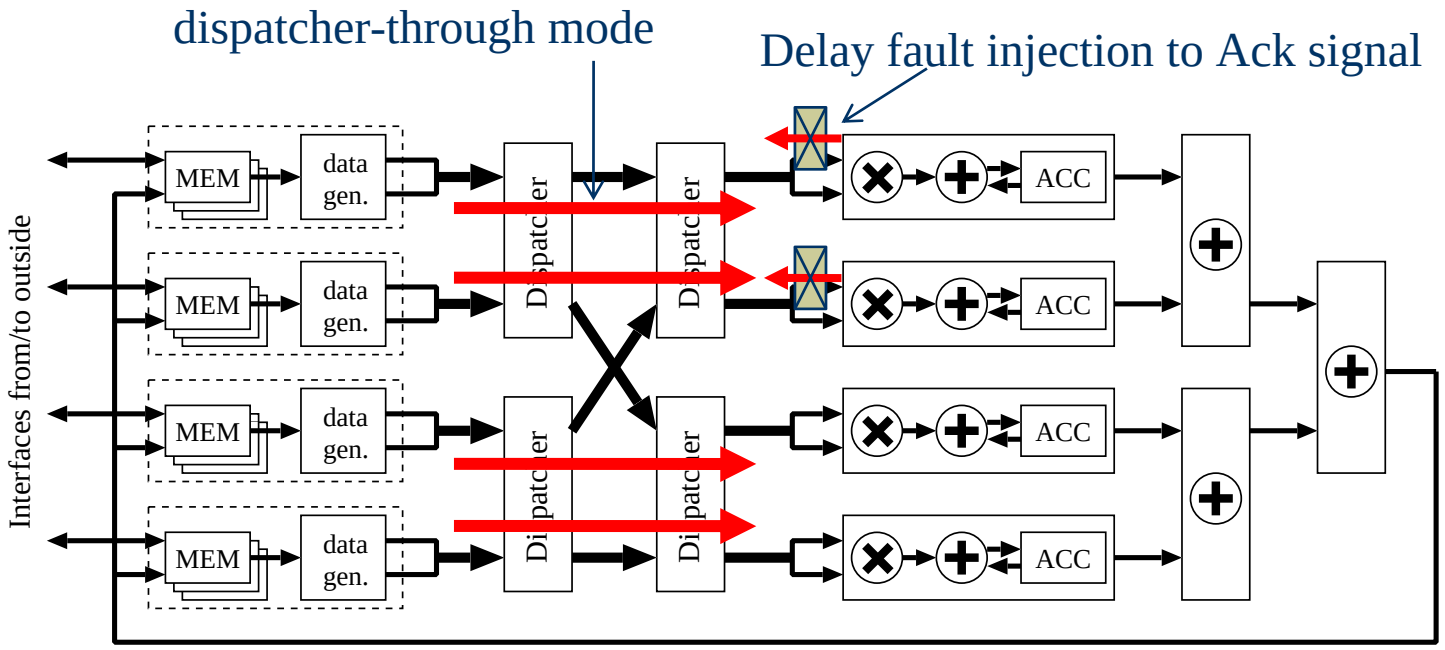
- ASIC implementation
  - 32 16-bit-variable solver using 0.13um process technology



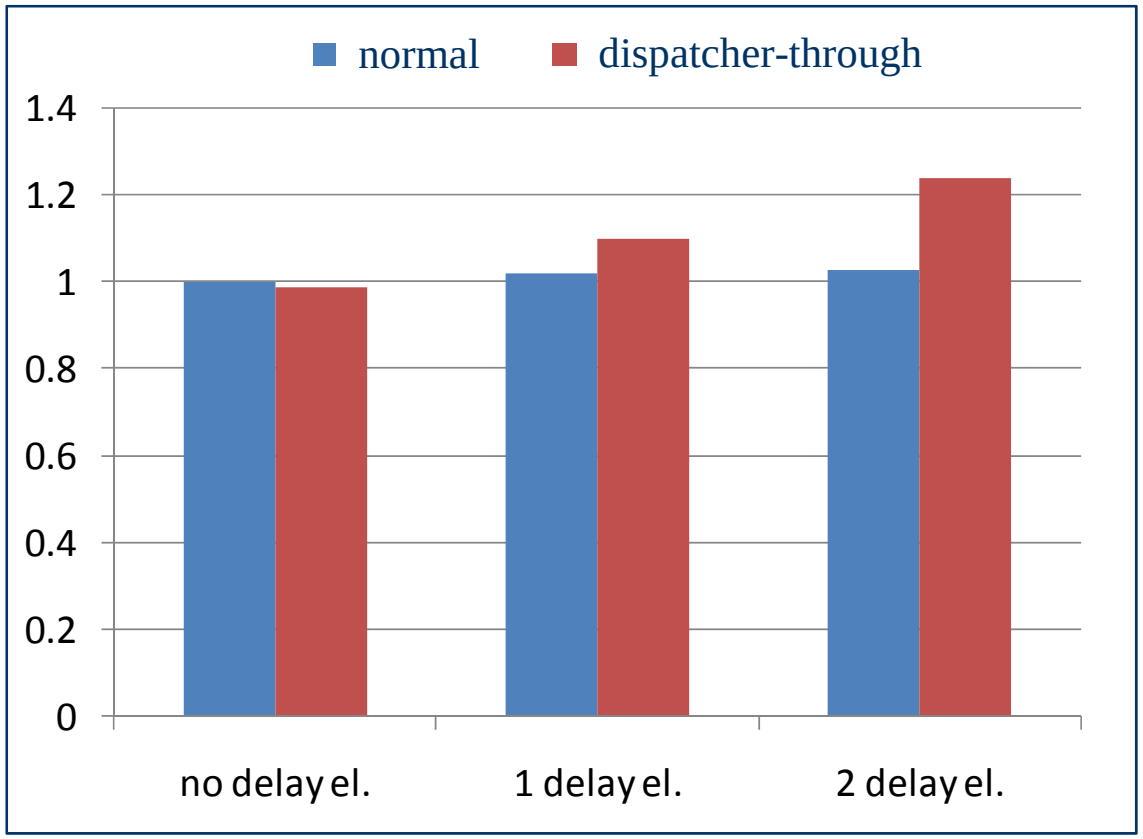


Experiments (2)

- Evaluation
  - Normal mode / dispatcher-through mode
  - Delay fault injection



Experiments (3)



Conclusion

- Propose a method to give degradation tolerance to a hardware accelerator with parallel functional units
  - Based on asynchronous circuit technology
  - Data-flow control based on completion signal
- A linear equation solver is implemented as a case study
  - Much better degradation tolerance is shown