An Approach to Tolerating Delay Faults based on Asynchronous Circuits

Tomohiro Yoneda
National Institute of Informatics

Masashi Imai — Univ. of Tokyo
Atsushi Matsumoto — Tohoku Univ.
Takahiro Hanyu — Tohoku Univ.
Yoichi Nakamura — NEC

Background and Goal

- Advances in semiconductor technologies
  - new types of faults
    - NBTI (Negative Bias Temperature Instability)
- Faults considered in this work
  - degradation of operational units by increased delays due to several effects
- Goal of this work
  - propose an approach to tolerating such "delay faults caused during operation"
  - using Asynchronous circuit technology

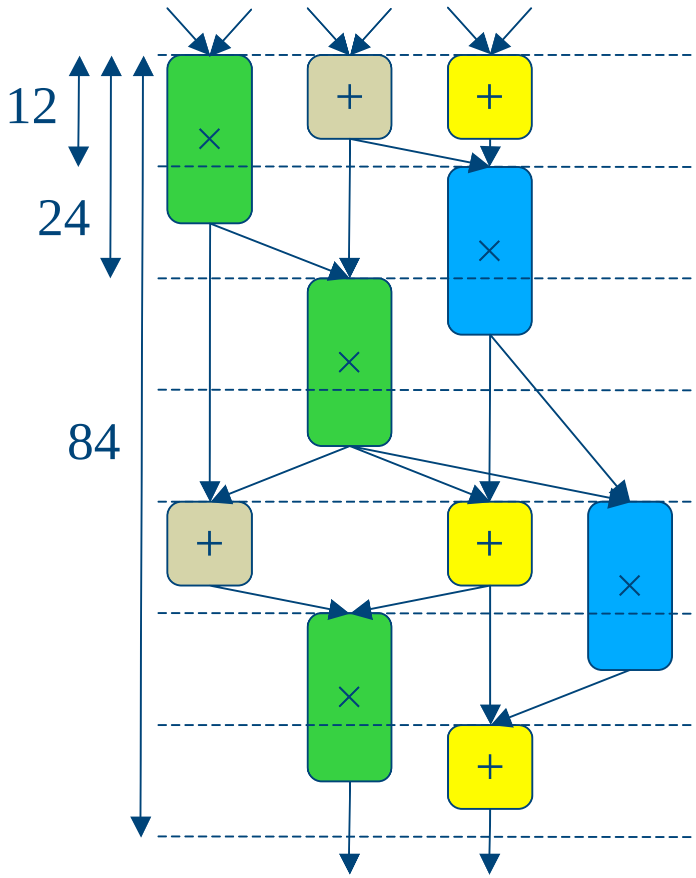
Framework

- Data flow graph level
  - e.g., hardware accelerators such as a DCT (Discrete Cosine Transform) module
  - contain several "basic operational units" such as adders, multipliers, etc.
- Assumptions
  - Each "occurrence of a delay fault" increases the delay of the affected basic op. unit
  - A hardware module has a "deadline" for the completion of the computation

Effect of delay faults (1)

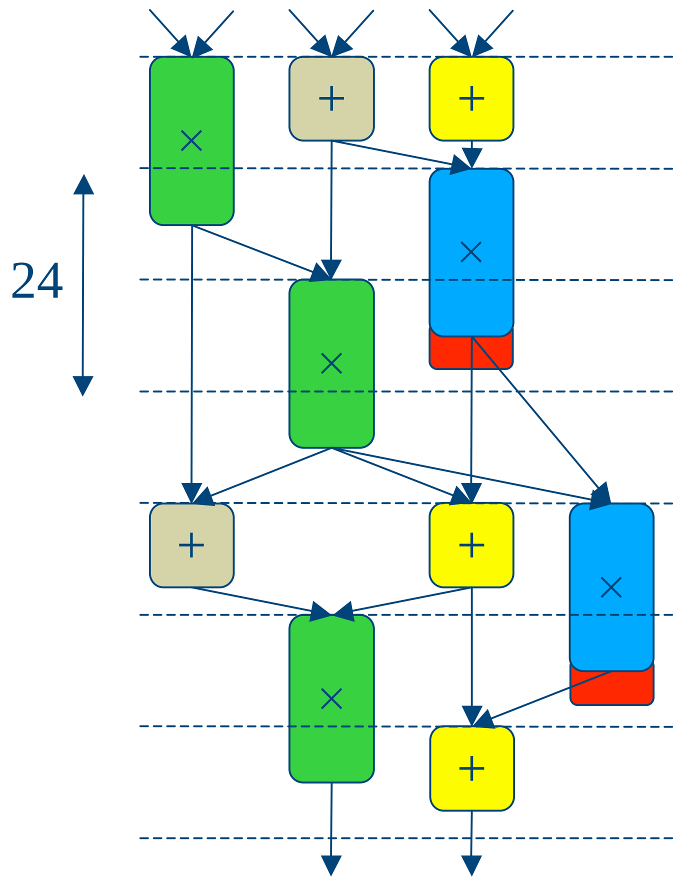
- A simple example
  - operational time
    - Multiplier: 20 (2 clk. op.)
    - Adder: 10
  - degradation
    - 15% for each delay fault occurrence
  - Margin: 20%
    - clock: 12
    - total: 84 (deadline)

2 Multipliers, 2 Adders



Effect of delay faults (2)

- In the case of a synchronous circuit



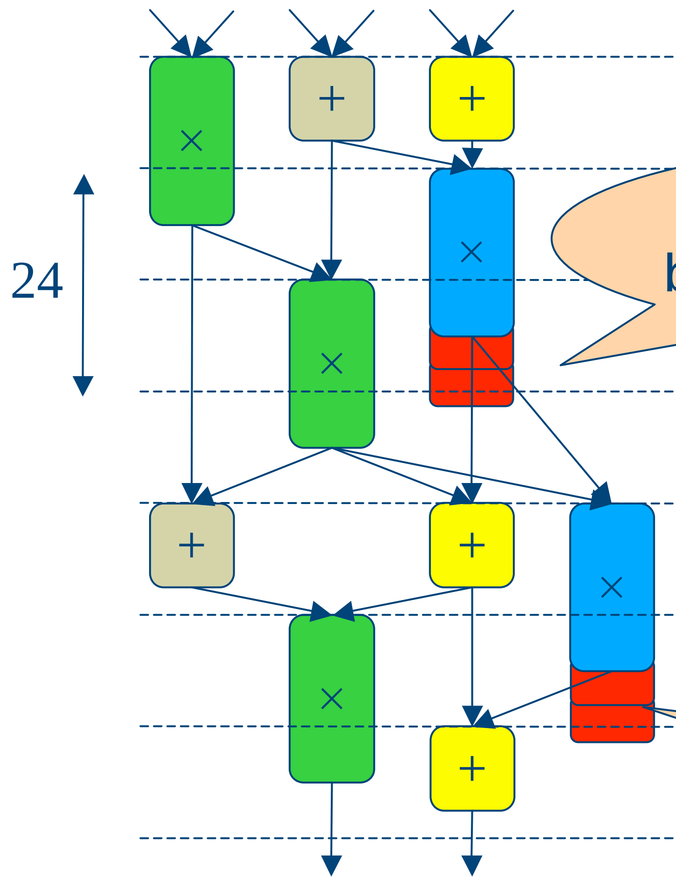
1. A delay fault at the blue unit
   - delay 20 → delay 23

Nothing happens

National Institute of Informatics
国立情報学研究所

Effect of delay faults (2)

- In the case of a synchronous circuit



1. A delay fault at the blue ... elay 23
2. ... fault at the same unit
   - delay 23 → delay 26

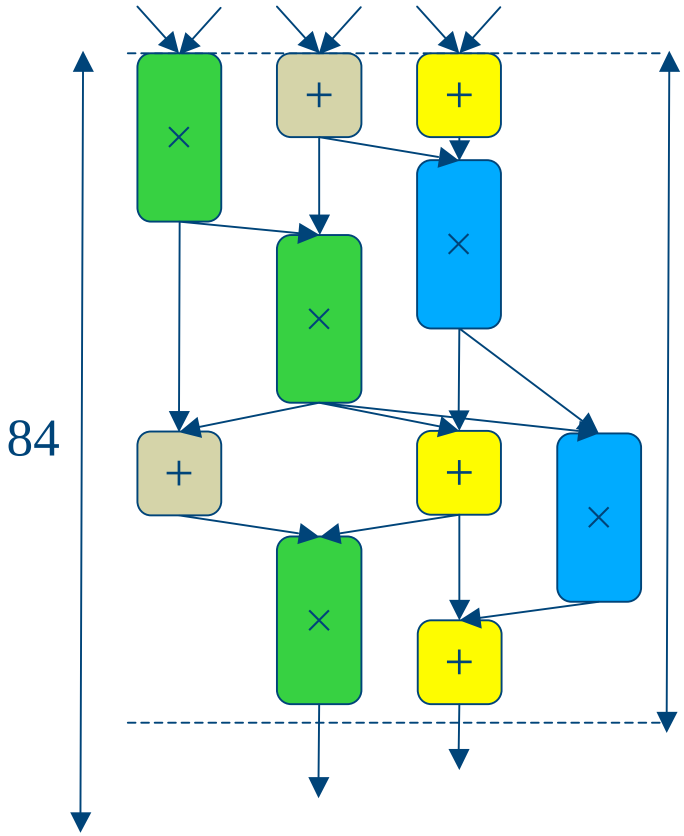
This may be avoided by delaying latch timing

System malfunctions!

This is hopeless because data is used in the next cycle

Effect of delay faults (3)

- In the case of an asynchronous circuit



10% of clock period for asynchronous circuit overhead

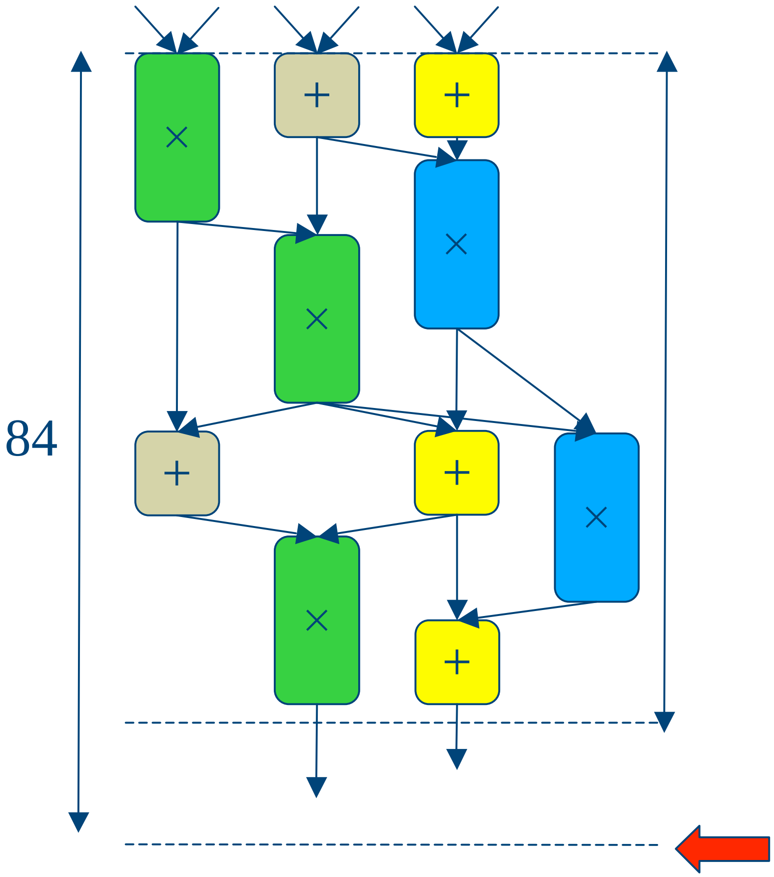
20 + 1 + 20 + 1 + 10 + 1 + 20 + 1
= 74

deadline of computation

National Institute of Informatics
国立情報学研究所

Effect of delay faults (3)

- In the case of an asynchronous circuit

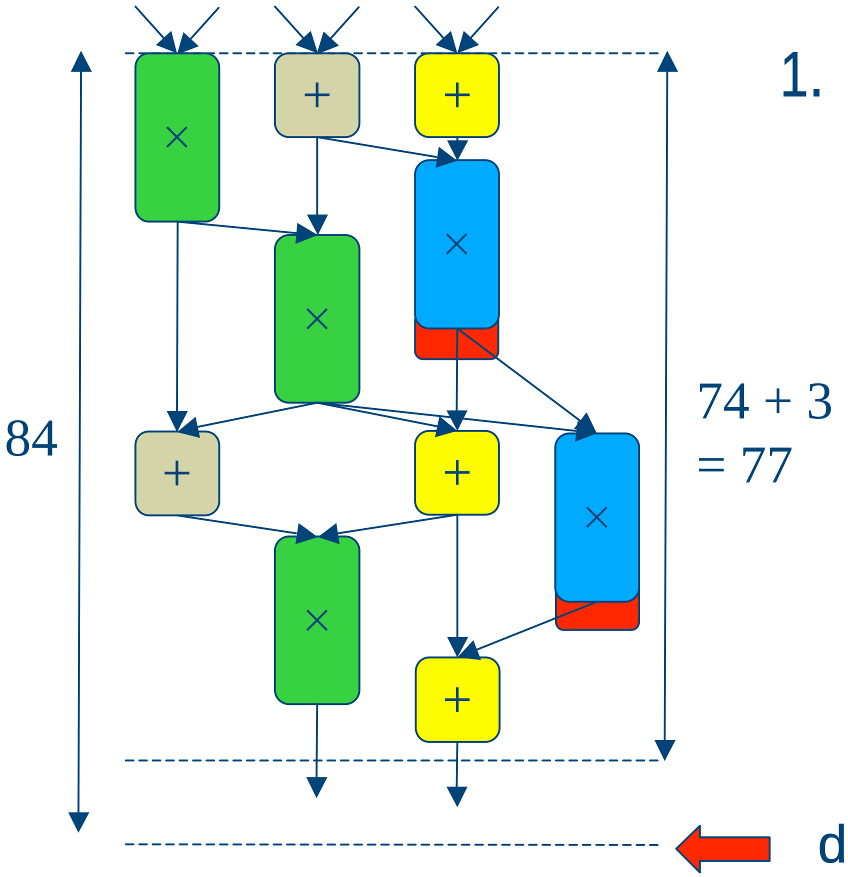


1. A delay fault at the blue unit
   - delay 20 → delay 23

deadline of computation

Effect of delay faults (3)

- In the case of an asynchronous circuit

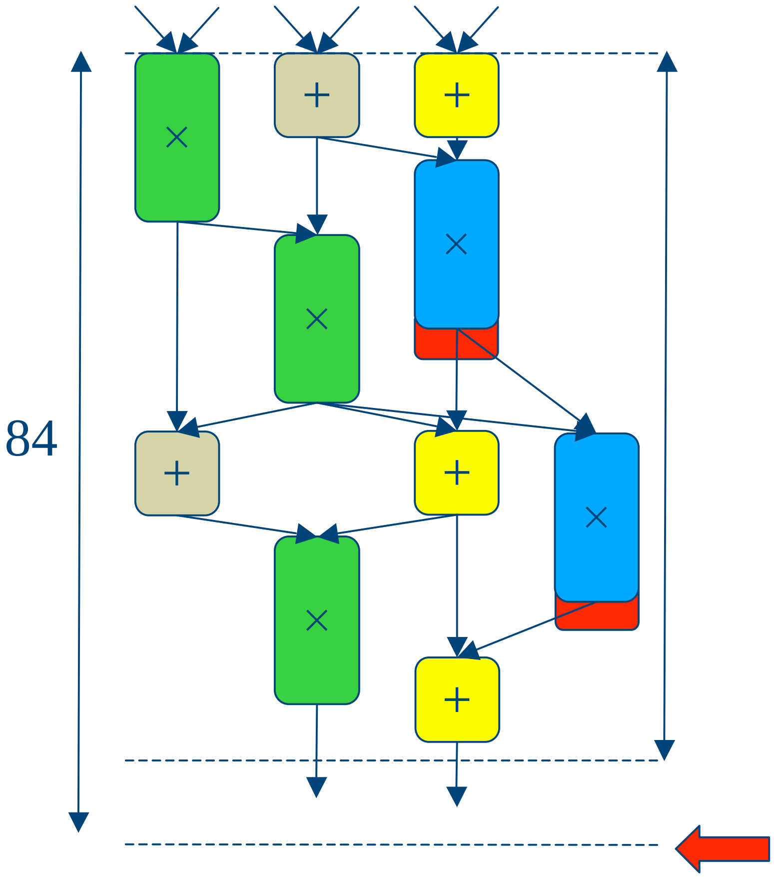


1. A delay fault at the blue unit
   - delay 20 → delay 23

deadline of computation

Effect of delay faults (3)

- In the case of an asynchronous circuit

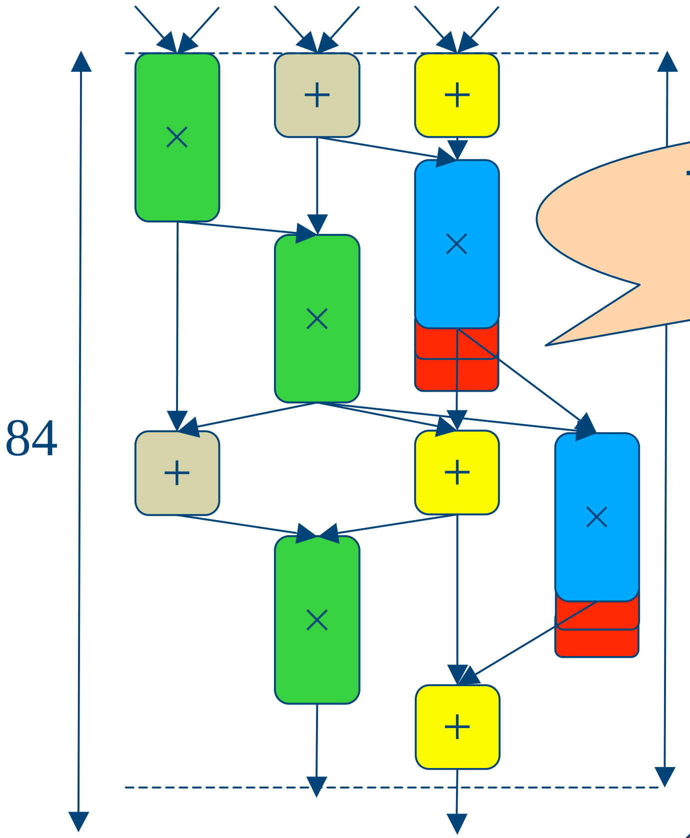


1. A delay fault at the blue unit
   - delay 20 → delay 23
2. Second delay fault at the same unit
   - delay 23 → delay 26

deadline of computation

Effect of delay faults (3)

- In the case of an asynchronous circuit



1. A delay fault at the blue

elay 23

delay fault at the same unit

- delay 23 → delay 26

Thanks to this slack, delays are absorbed

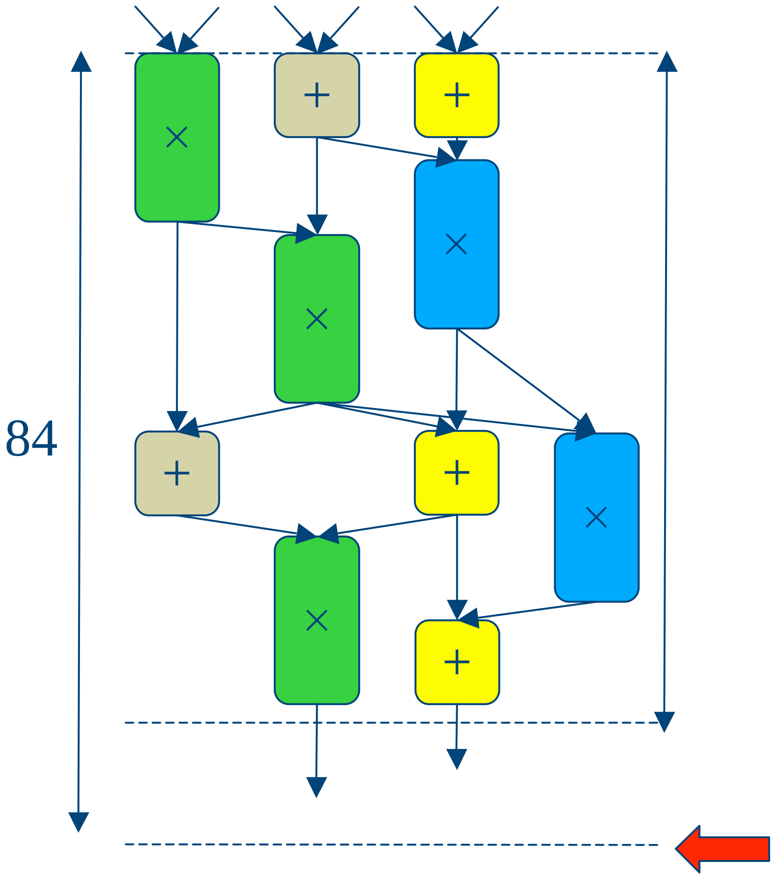
74 + 3 + 3 = 80

System still works!

deadline of computation

Effect of delay faults (4)

- In the case of an asynchronous circuit

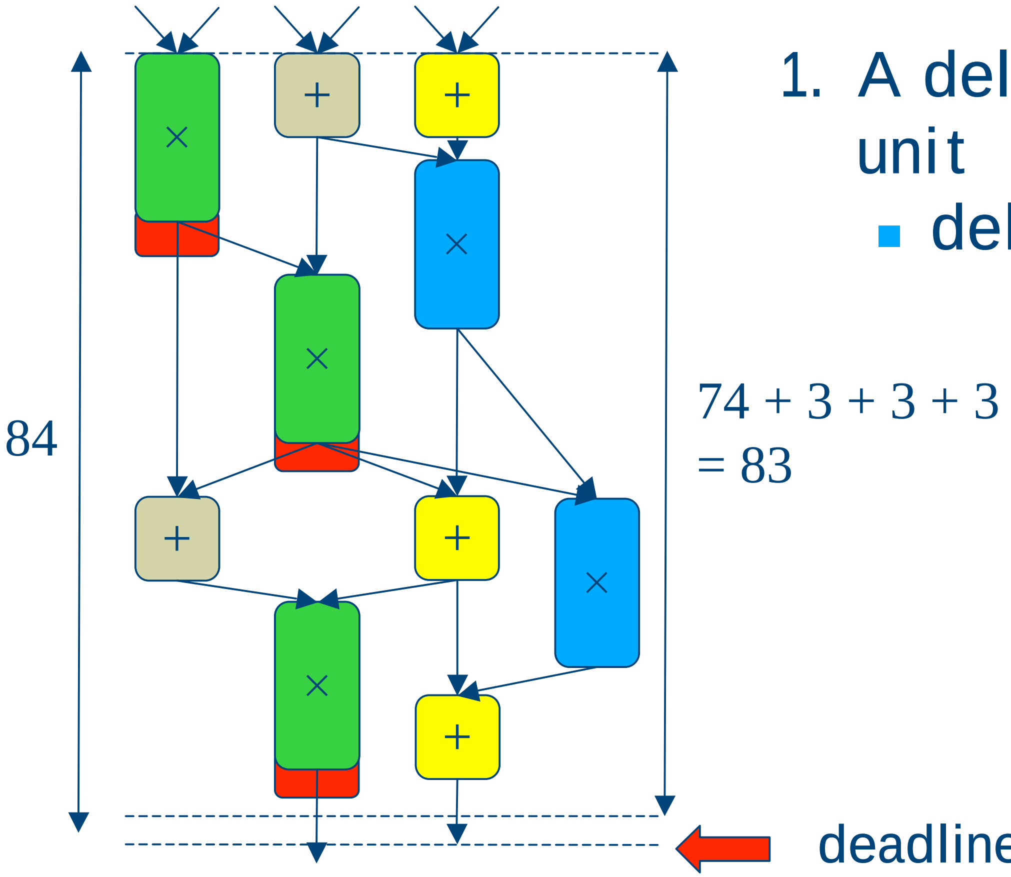


1. A delay fault at the green unit
   - delay 20 → delay 23

deadline of computation

Effect of delay faults (4)

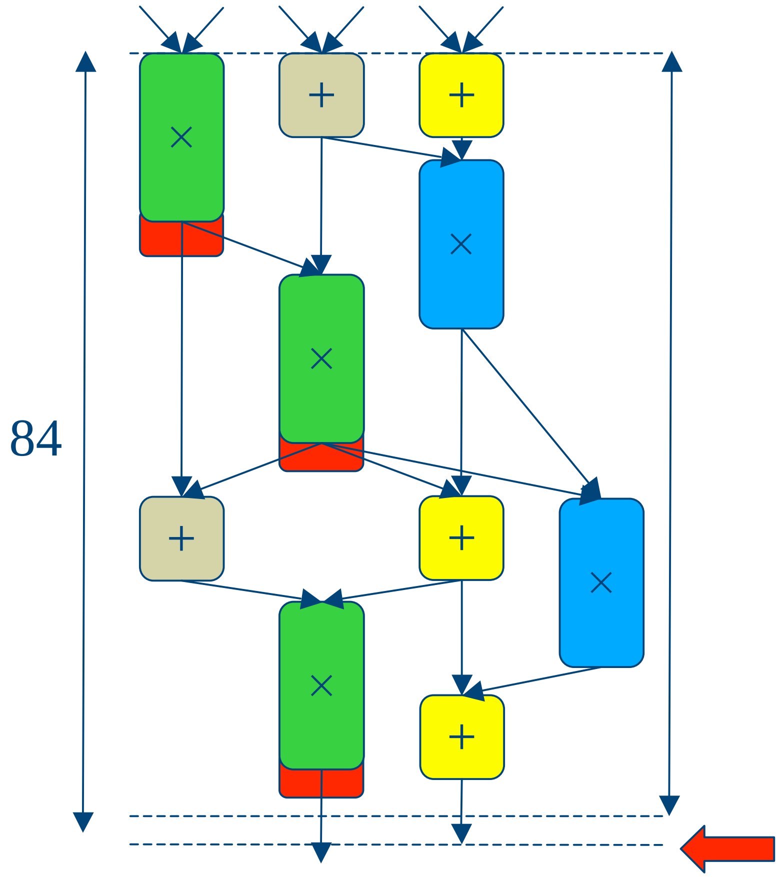
- In the case of an asynchronous circuit



1. A delay fault at the green unit
   - delay 20 → delay 23

Effect of delay faults (4)

- In the case of an asynchronous circuit

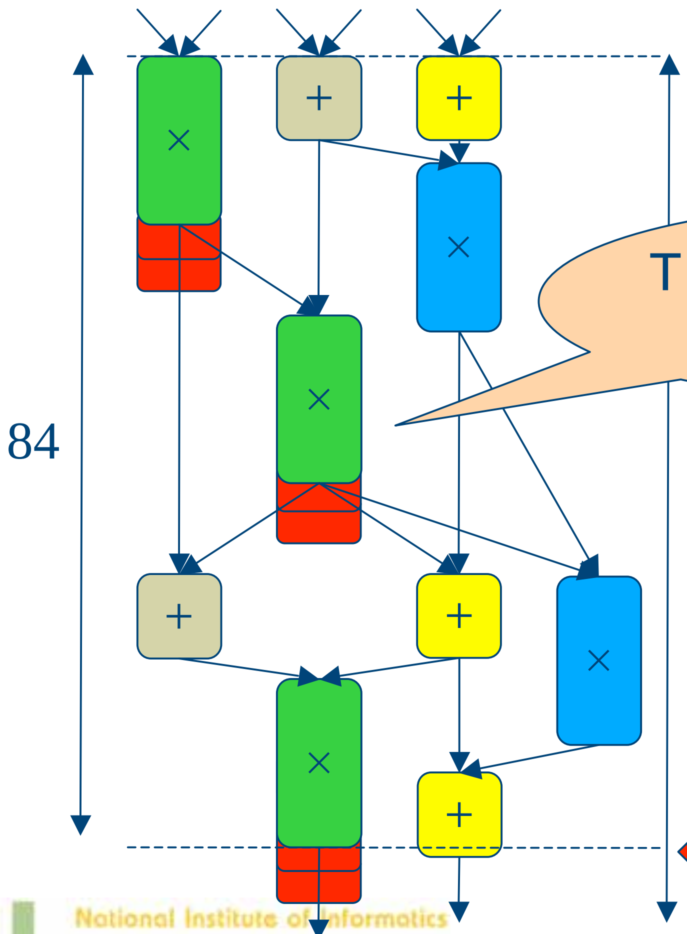


1. A delay fault at the green unit
   - delay 20 → delay 23
2. Second delay fault at the same unit
   - delay 23 → delay 26

deadline of computation

Effect of delay faults (4)

- In the case of an asynchronous circuit



1. A delay fault at the green unit
   - delay 20 → delay 23
2. fault at the same unit
   - delay 23 → delay 26

This green unit has NO SLACK

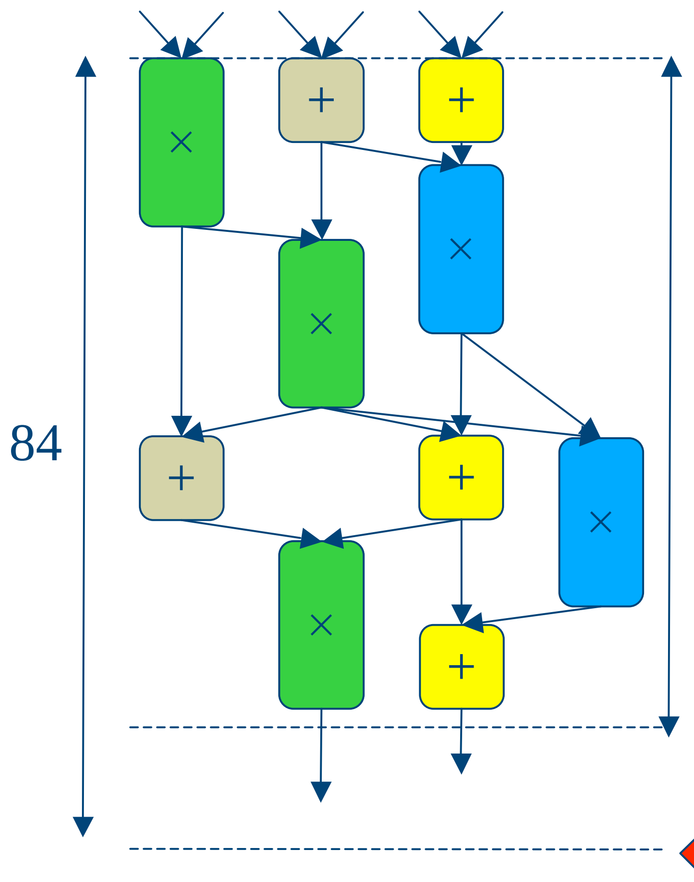
System fails in producing outputs in time!

74 + 6 + 6 + 6
= 92

deadline of computation

Idea for tolerating delay faults

- “Detect” the affected unit, and “reallocate” the related operations

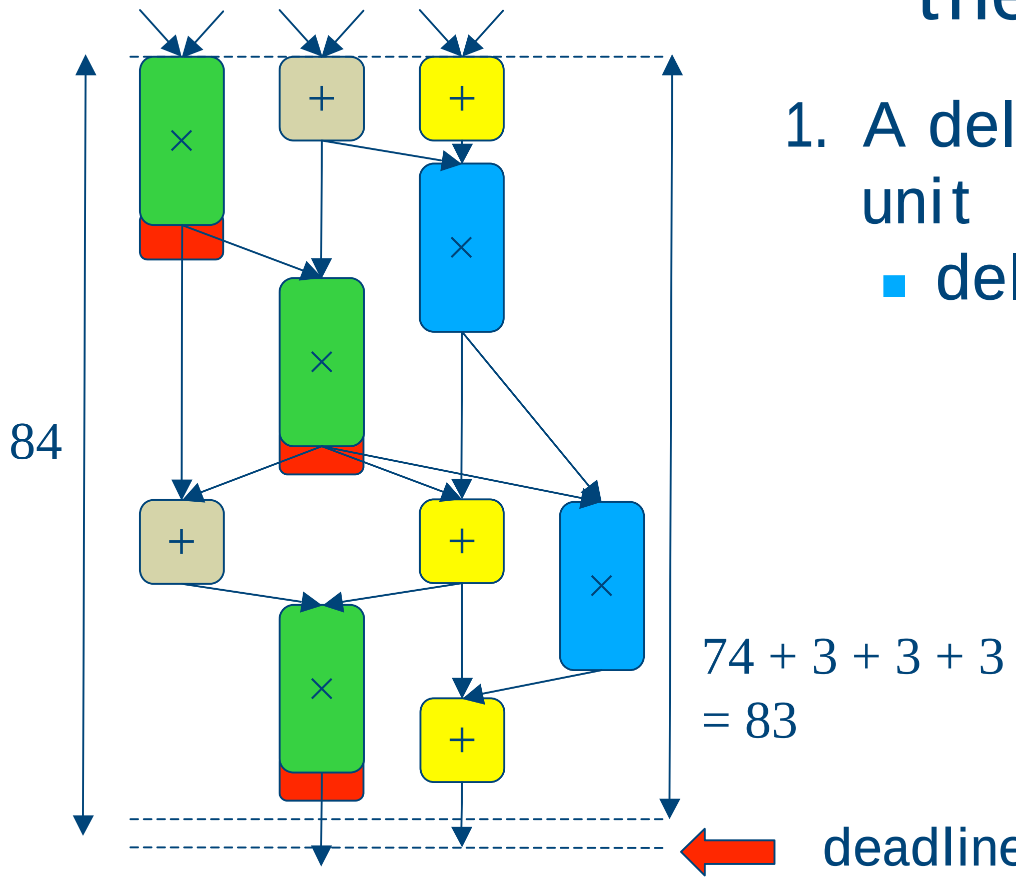


1. A delay fault at the green unit
   - delay 20 → delay 23

deadline of computation

Idea for tolerating delay faults

- "Detect" the affected unit, and "reallocate" the related operations

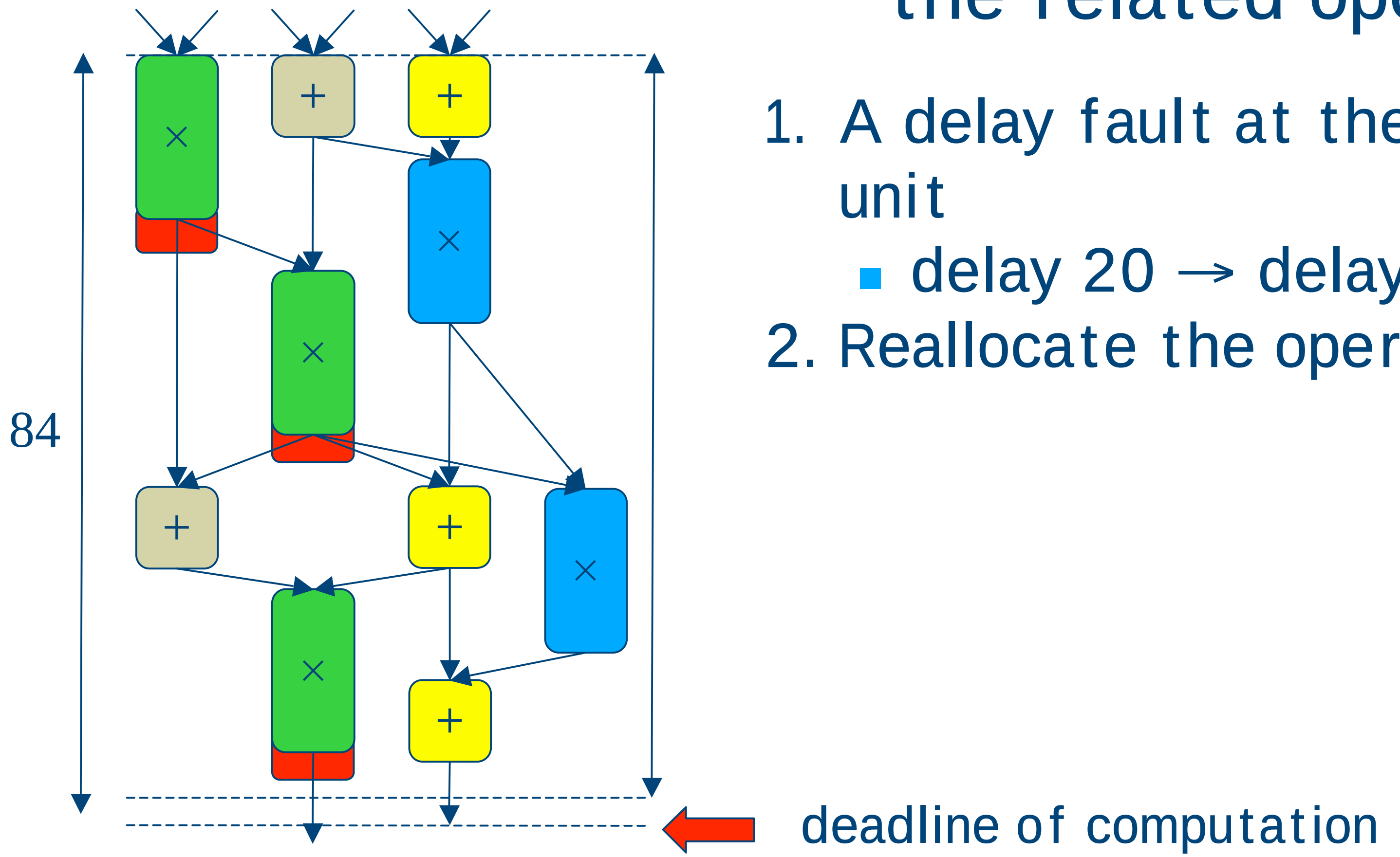


1. A delay fault at the green unit
   - delay 20 → delay 23

deadline of computation

Idea for tolerating delay faults

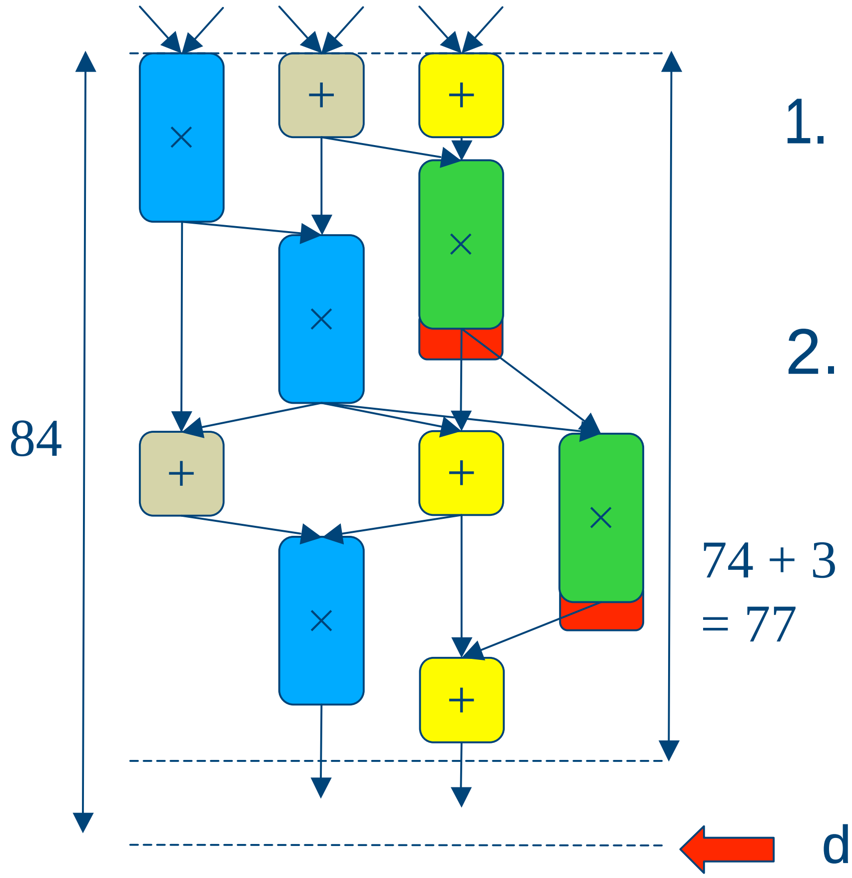
- "Detect" the affected unit, and "reallocate" the related operations



1. A delay fault at the green unit
   - delay 20 → delay 23
2. Reallocate the operations

Idea for tolerating delay faults

- "Detect" the affected unit, and "reallocate" the related operations

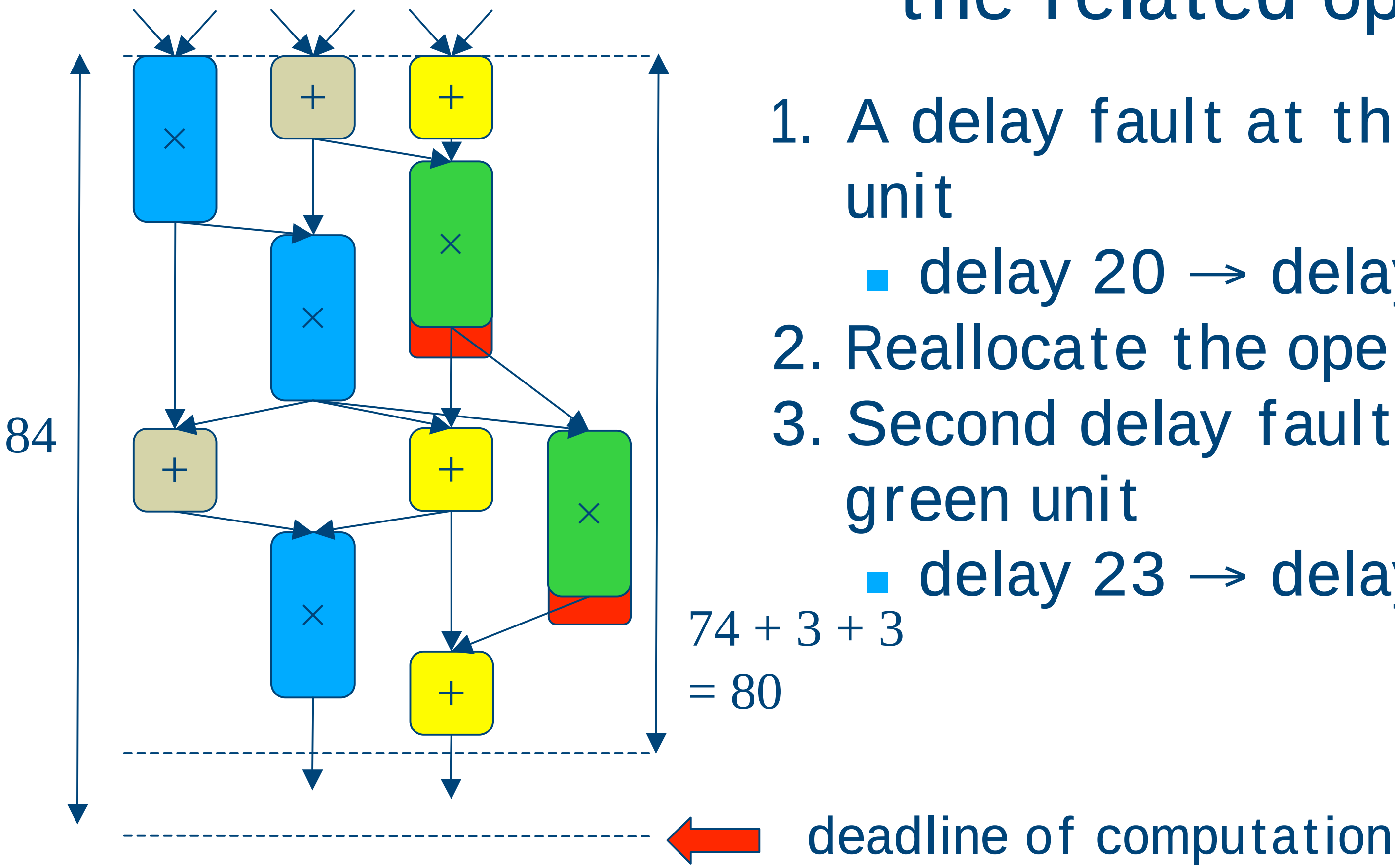


1. A delay fault at the green unit
   - delay 20 → delay 23
2. Reallocate the operations

deadline of computation

Idea for tolerating delay faults

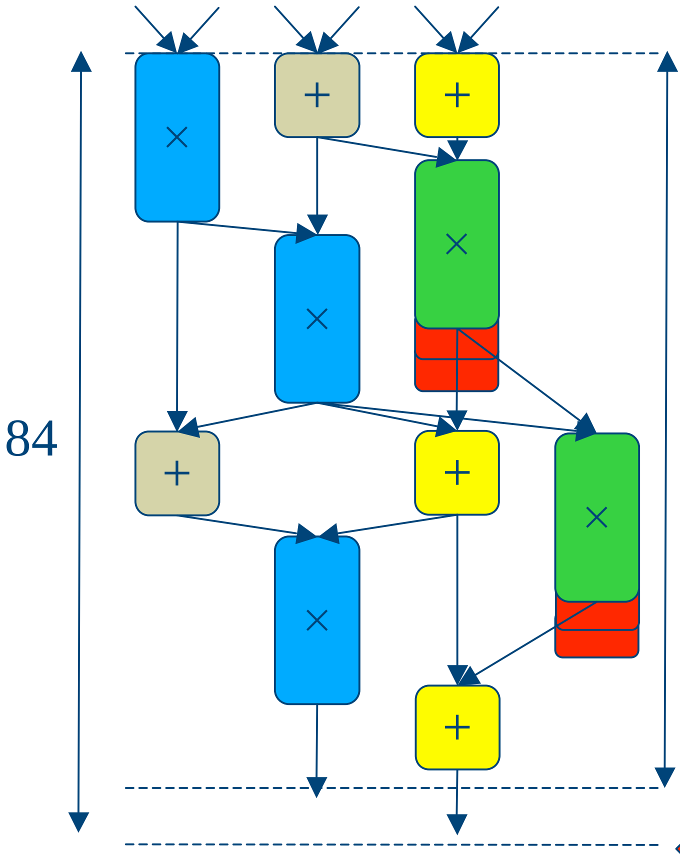
- "Detect" the affected unit, and "reallocate" the related operations



1. A delay fault at the green unit
   - delay 20 → delay 23
2. Reallocate the operations
3. Second delay fault at the green unit
   - delay 23 → delay 26

Idea for tolerating delay faults

- "Detect" the affected unit, and "reallocate" the related operations



1. A delay fault at the green unit
   - delay 20 → delay 23
2. Reallocate the operations
3. Second delay fault at the green unit
   - delay 23 → delay 26

74 + 3 + 3
= 80

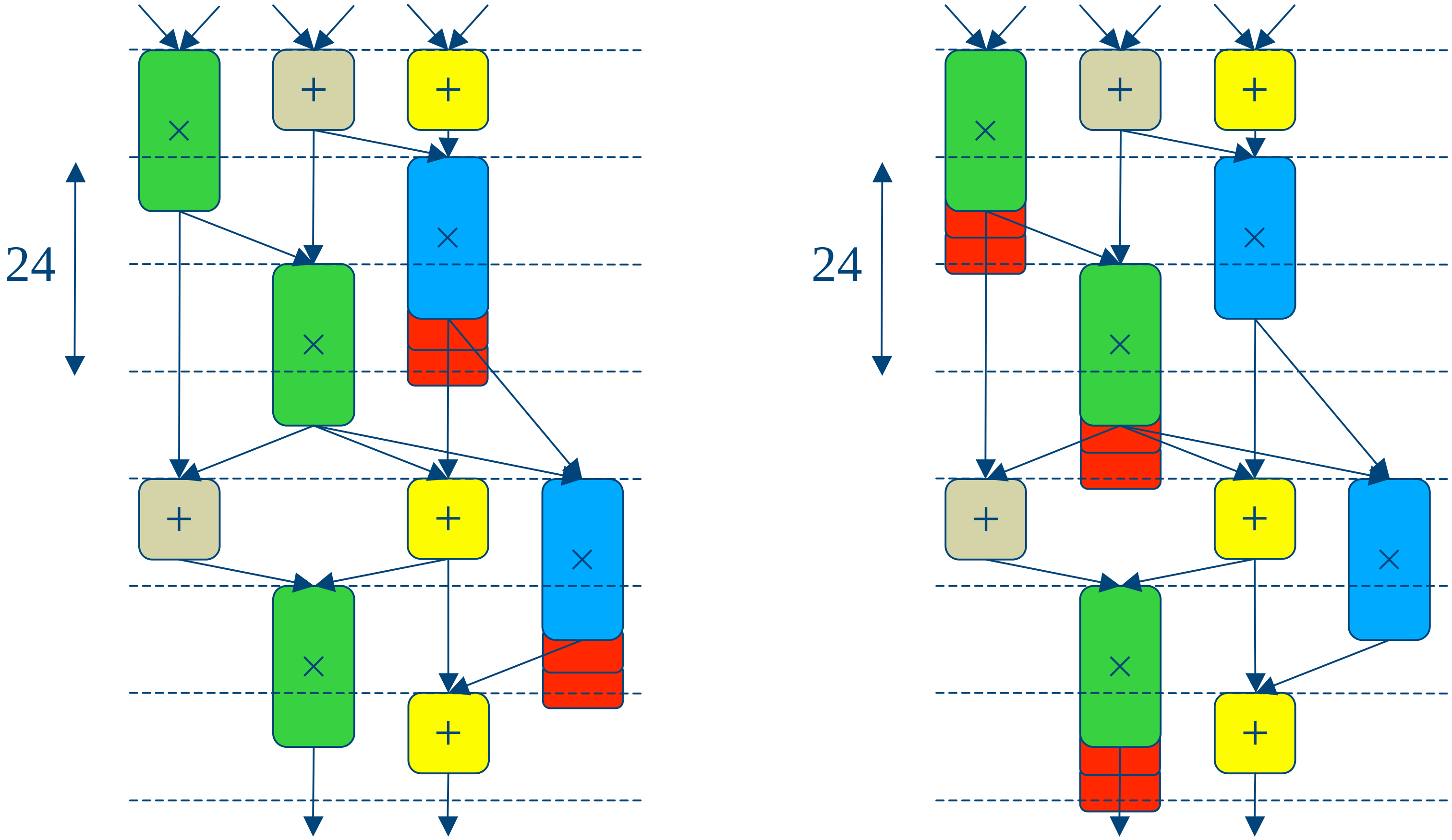
Double delay faults at green unit can be tolerated!

deadline of computation

National Institute of Informatics 国立情報学研究所

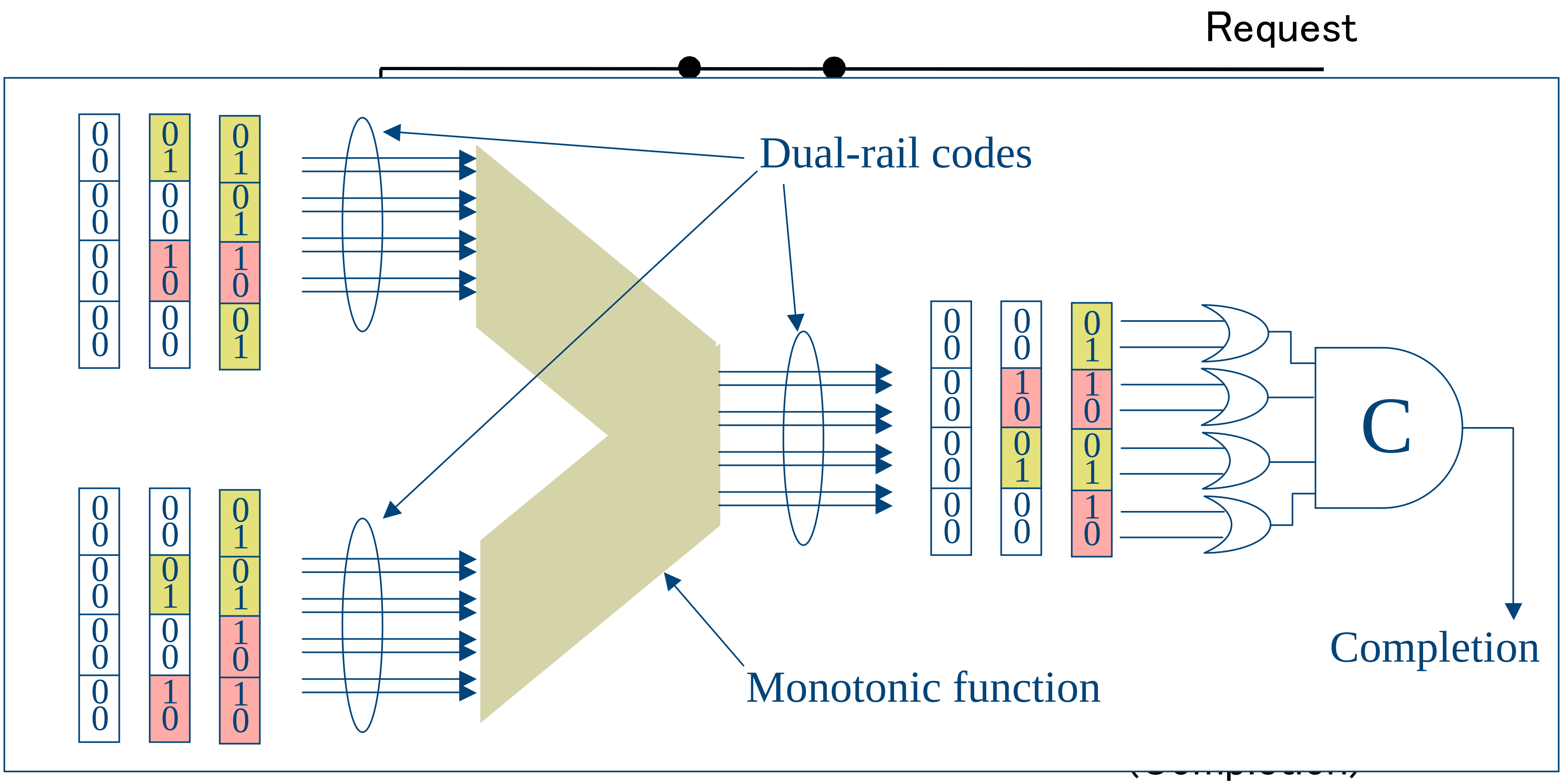
Idea for tolerating delay faults

- This idea doesn't work for sync. circuits



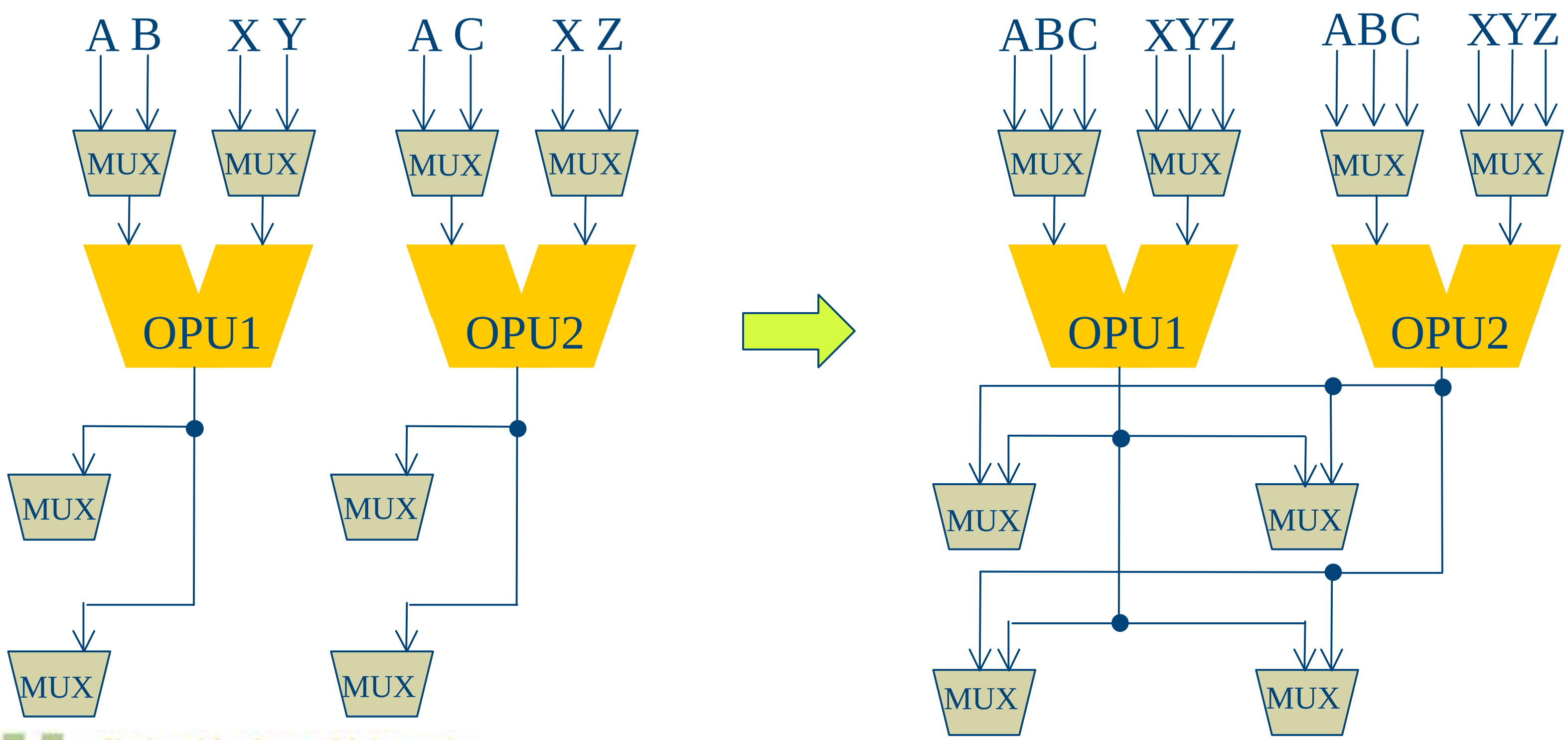
NII National Institute of Informatics 国立情報学研究所

Detection of a delay fault



Implementation of reallocation

- Larger freedom makes reliability higher, but implementation more complicated
  - Pair op. units and swap them

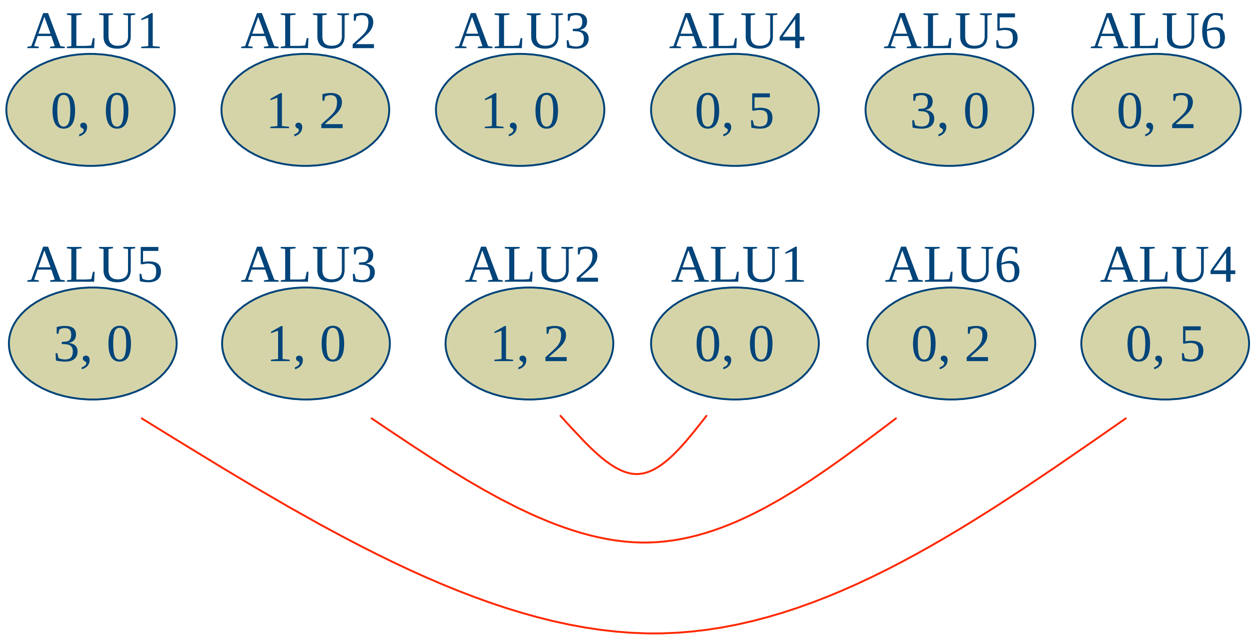


How to decide paired units (1)

- Slacks
  - ASAP time for an operation
    - Time to start the operation, when it is scheduled as soon as possible
  - ALAP time for an operation
    - Time to start the operation, when it is scheduled as latest as possible without delaying the completion
  - Slack for an operation =(ALAP time) – (ASAP time)
  - Slack for a unit = minimal value among the slacks for the operations performed by the unit
- Influence
  - Influence = (Completion time under a delay fault) – (Original completion time)

How to decide paired units (2)

- Pairing units with different (slack, influence) values
- Ex.



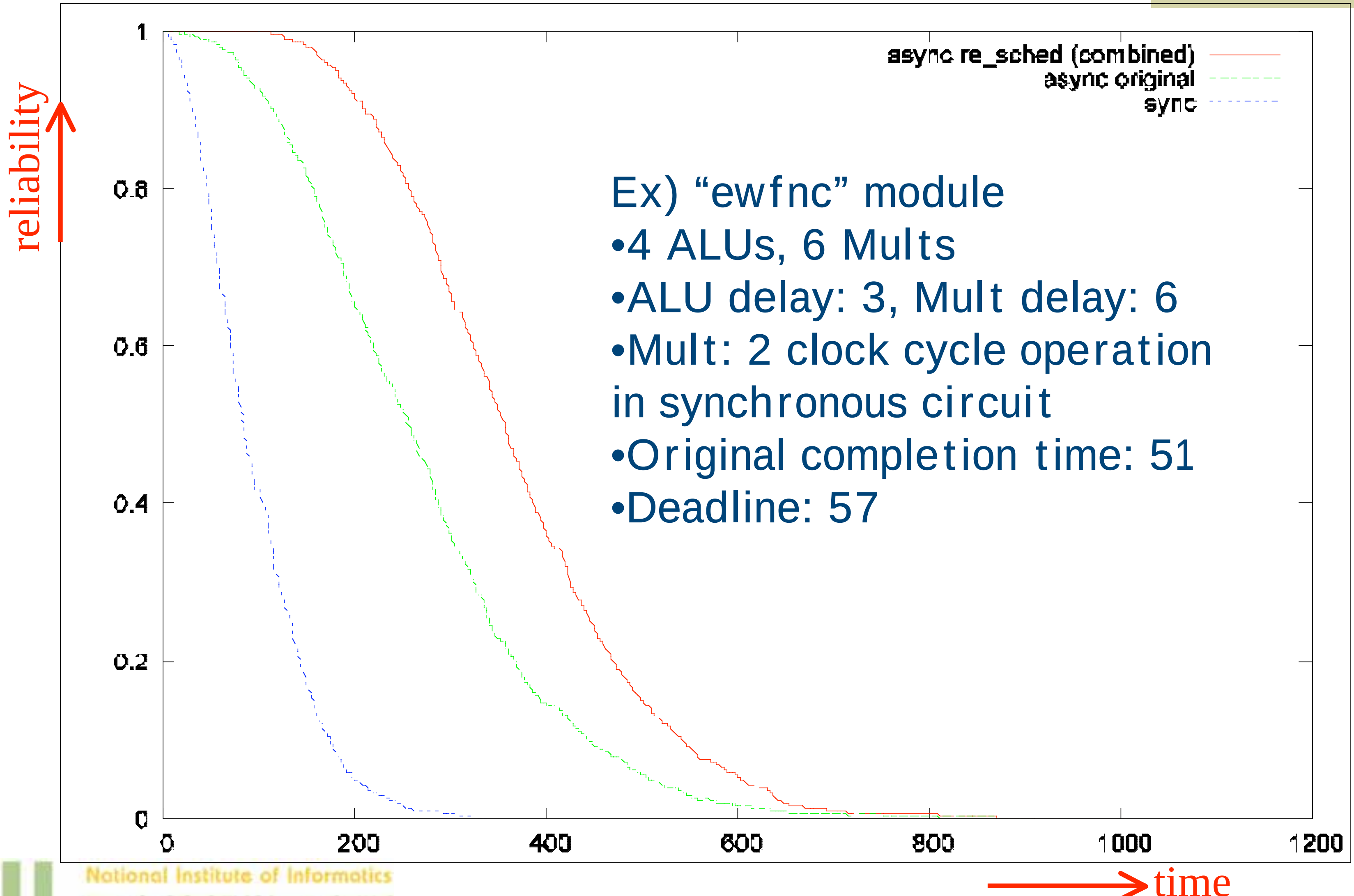
Evaluation (1)

- Delay fault injection
  - At every time step, a delay fault is injected in a fixed rate depending on total area
  - Each fault hits a randomly selected basic operational unit
    - Larger units have greater probability
  - A unit hit by a delay fault increases its delay by a fixed amount
    - delay faults accumulate in each unit

Evaluation (2)

- A computation module is “down”, if
  - Synchronous case: delay of any basic operational unit exceeds the clock period
  - Asynchronous case: total computation time exceeds the given deadline

Reliability improvement

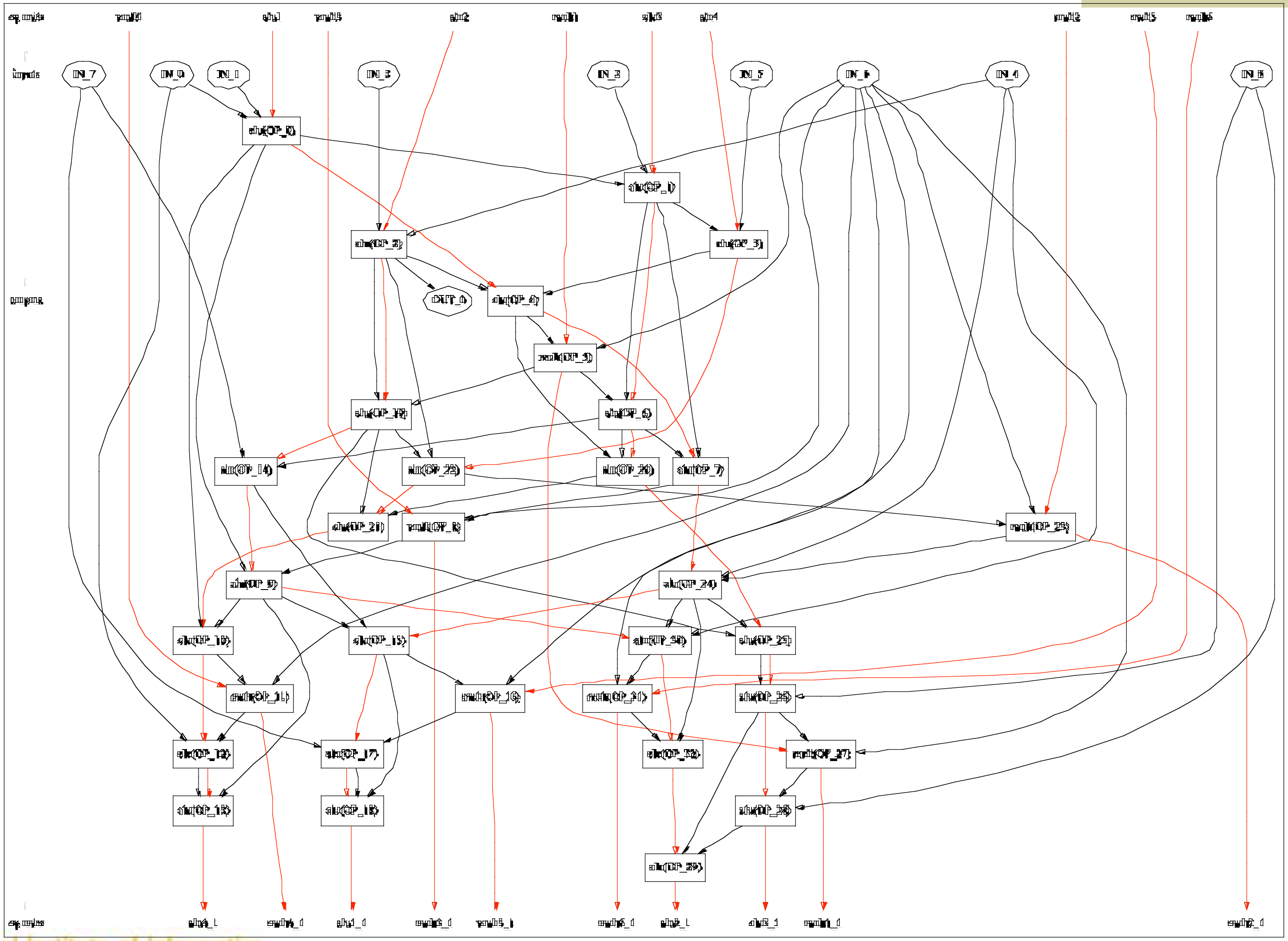


Ex) "ewfnc" module

- 4 ALUs, 6 Mults
- ALU delay: 3, Mult delay: 6
- Mult: 2 clock cycle operation in synchronous circuit
- Original completion time: 51
- Deadline: 57

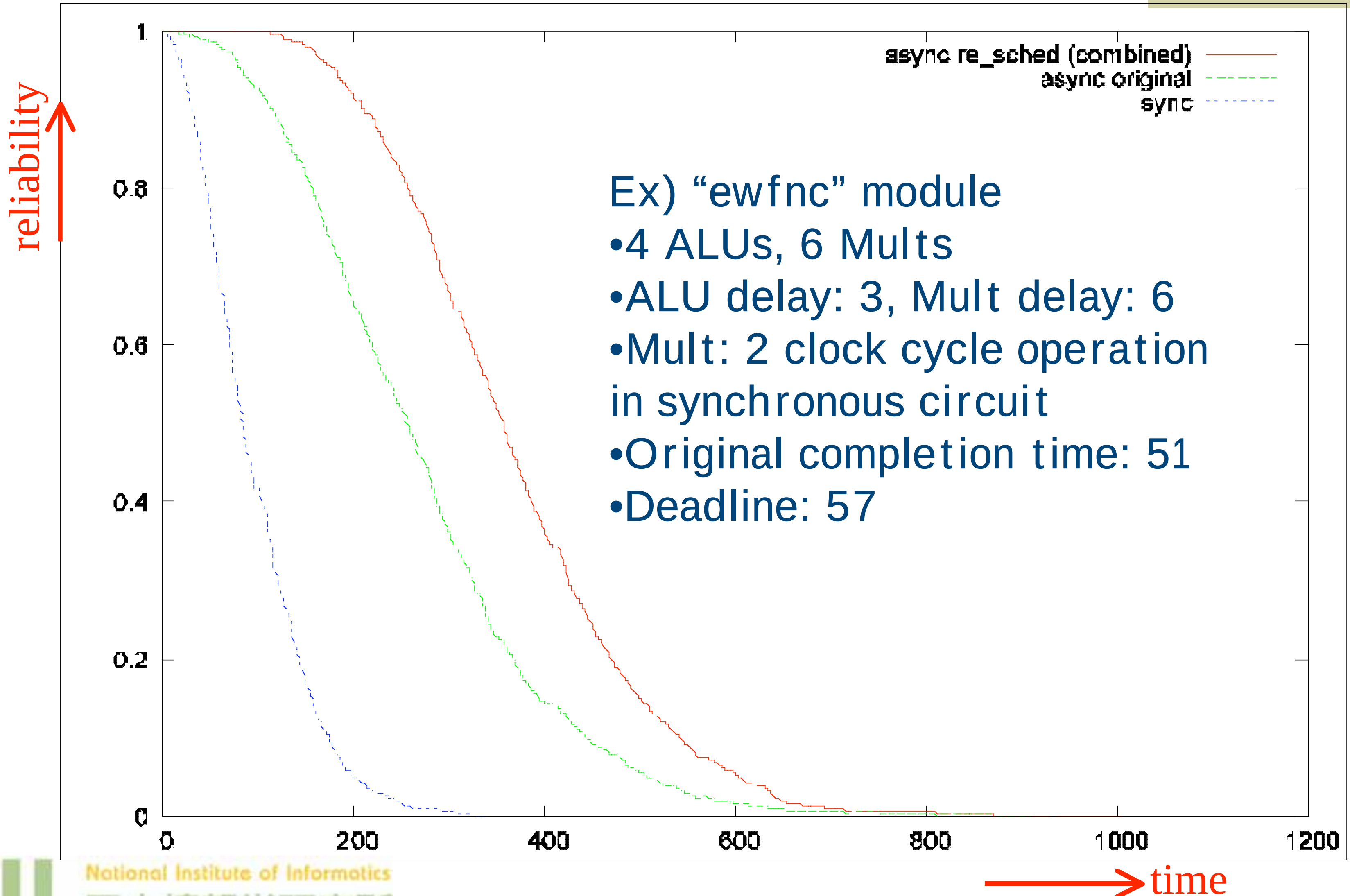
National Institute of Informatics
国立情報学研究所

Data flow graph of “ewfnc”



National Institute of Informatics
国立情報学研究所

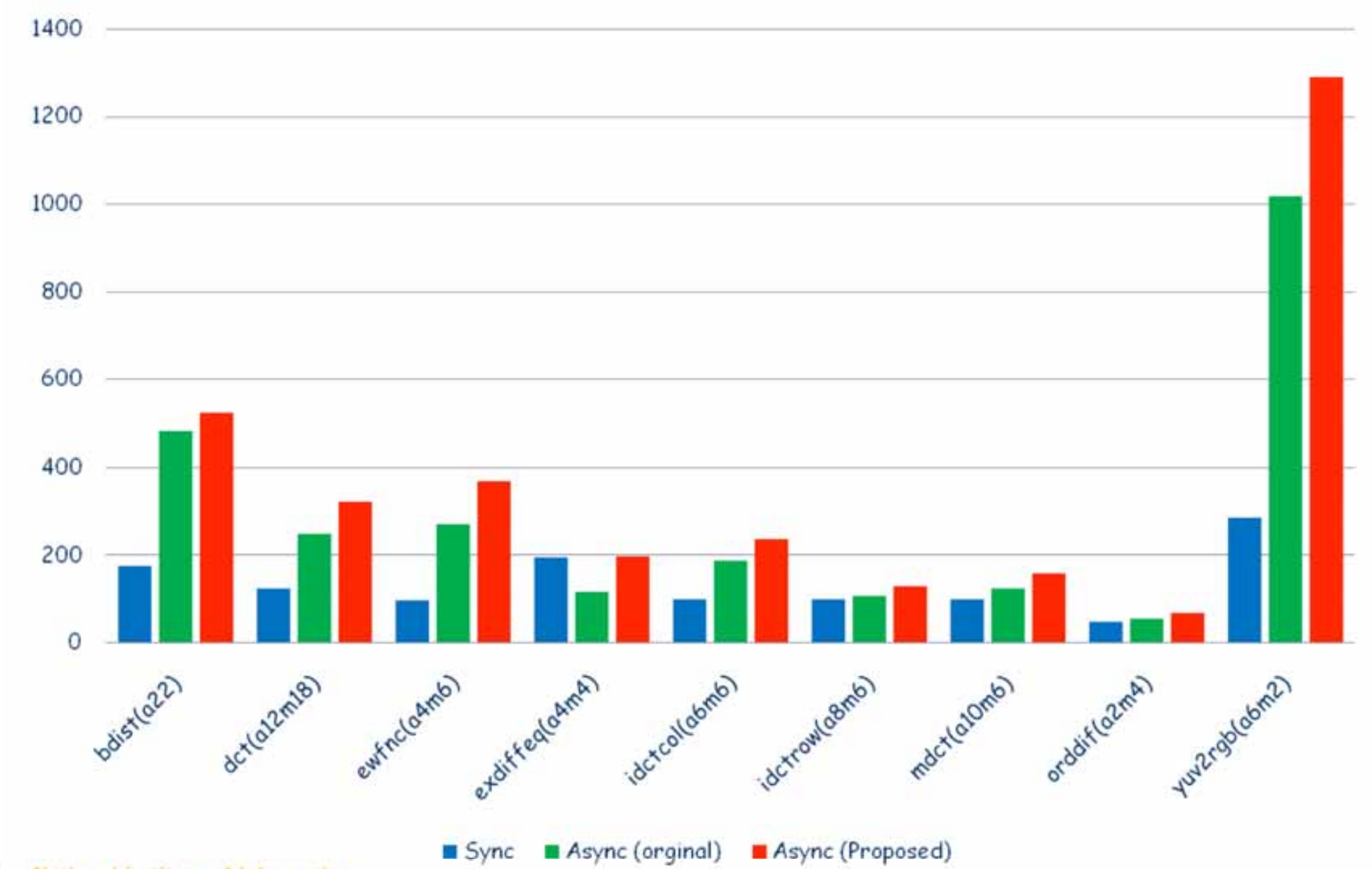
Reliability improvement



Ex) “ewfnc” module

- 4 ALUs, 6 Mults
- ALU delay: 3, Mult delay: 6
- Mult: 2 clock cycle operation in synchronous circuit
- Original completion time: 51
- Deadline: 57

MTTF improvement



National Institute of Informatics
国立情報学研究所

Conclusion

- An approach to tolerating delay faults based on
  - asynchronous circuits with coded data path
  - swapping mechanism with paired operational units
- Future work
  - extension for handling stuck-at-faults