FPGA Implementation Of Diffusive Realization For A Distributed Control Operator

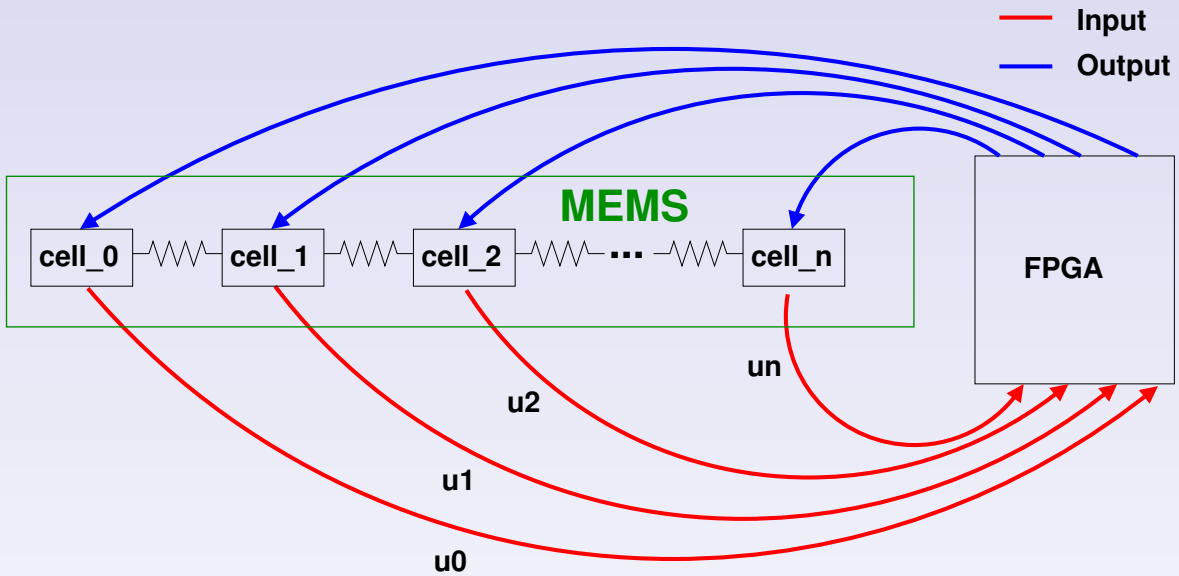
Gwenhaël Goavec-Merou[1 2], Youssef Yakoubi[3], Raphaël Couturier[4], Michel Lenczner[1], Jean-Michel Friedt[1] and F. Yang[5]

[1]FEMTO-ST, Time-Frequency Department, 26, chemin de l'Epitaphe, 25030 Besançon, France
[2]ARMadeus Systems 40, rue Marc Seguin, 68200 Mulhouse, France
[3]Laboratoire Jacques-Louis Lions, Pierre et Marie Curie University, 75252 Paris Cedex 05, France
[4]University of Franche-Comte, LIFC, IUT Belfort-Montbéliard, rue Engel Gros, 90000 Belfort, France
[5]Le2i, Bourgogne University, Mirande University, 21000 Dijon, France

June 29, 2010

Context

Real time control for distributed MEMS: heat equation in one-dimension

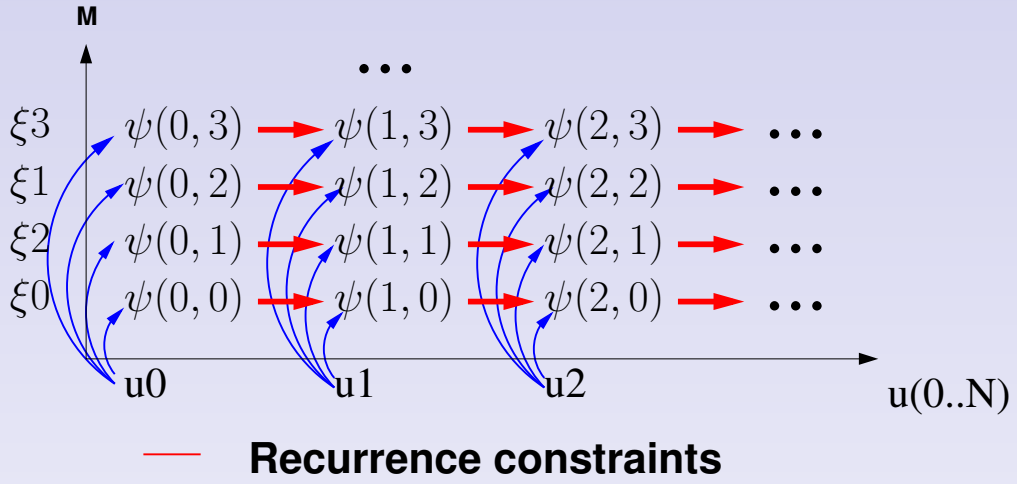


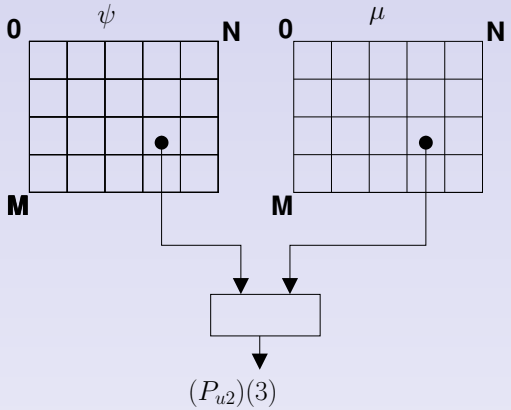
Goal: maximizing aquisition bandwith (10kHz to 1MHz) and maintain linear scalability with respect to network size.

Plan

1. Equations and Constraints
2. Analysis of the Problem
3. Implementation
4. Experimental Results

Equations





- All values are of Complex type.
- α_k, γ_k, β_k and $\mu_k(x_n)$ are preprocessed on a computer and transmitted to the FPGA.

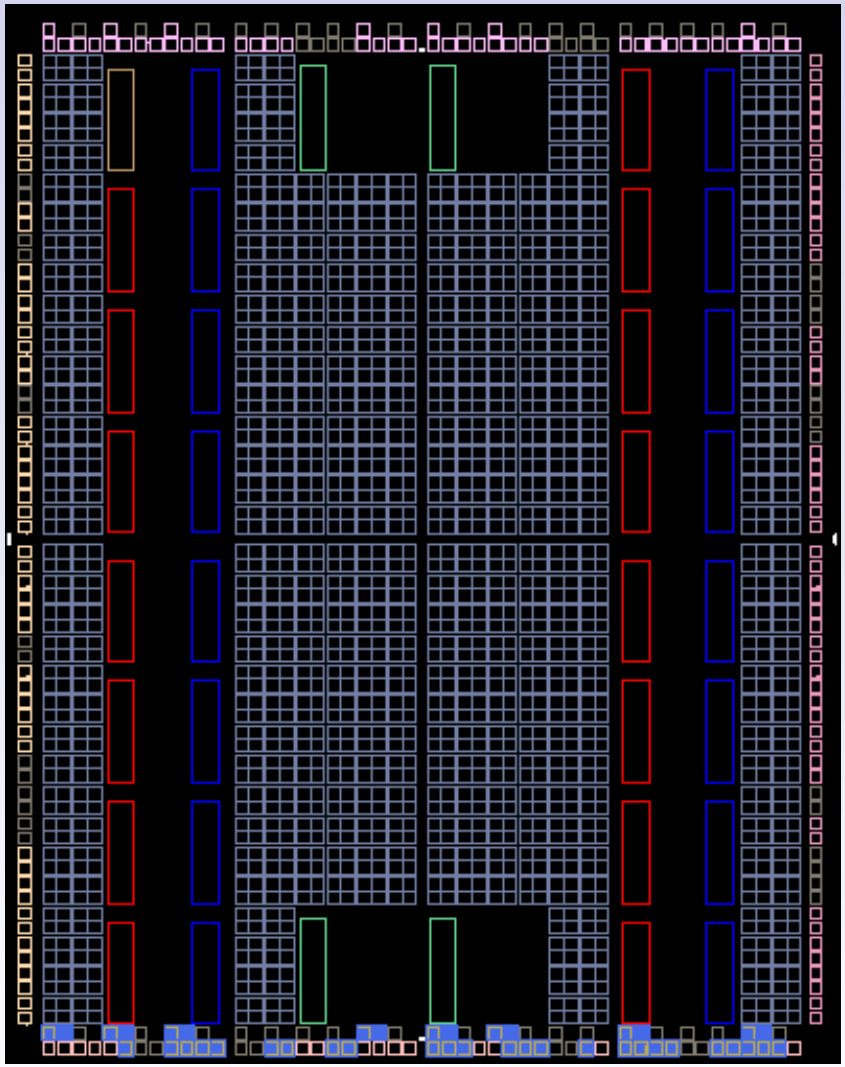
$$\psi(u)(x_{n+1}, \xi_k) = \alpha_k \psi(u)(x_n, \xi_k) + \beta_k u(x_n)$$

$$(Pu)(x_n) = \sum_{k=-M}^{M} (\alpha_k \psi(u)(x_n, \xi_k) + \gamma_k u(x_n))\mu_k(x_n)$$

Hardware Constraints

Xilinx Spartan 3A:

- 100MHz clock,
- 200K gates,
- 16 block multipliers,
- 16 RAMs (16kB/RAM).

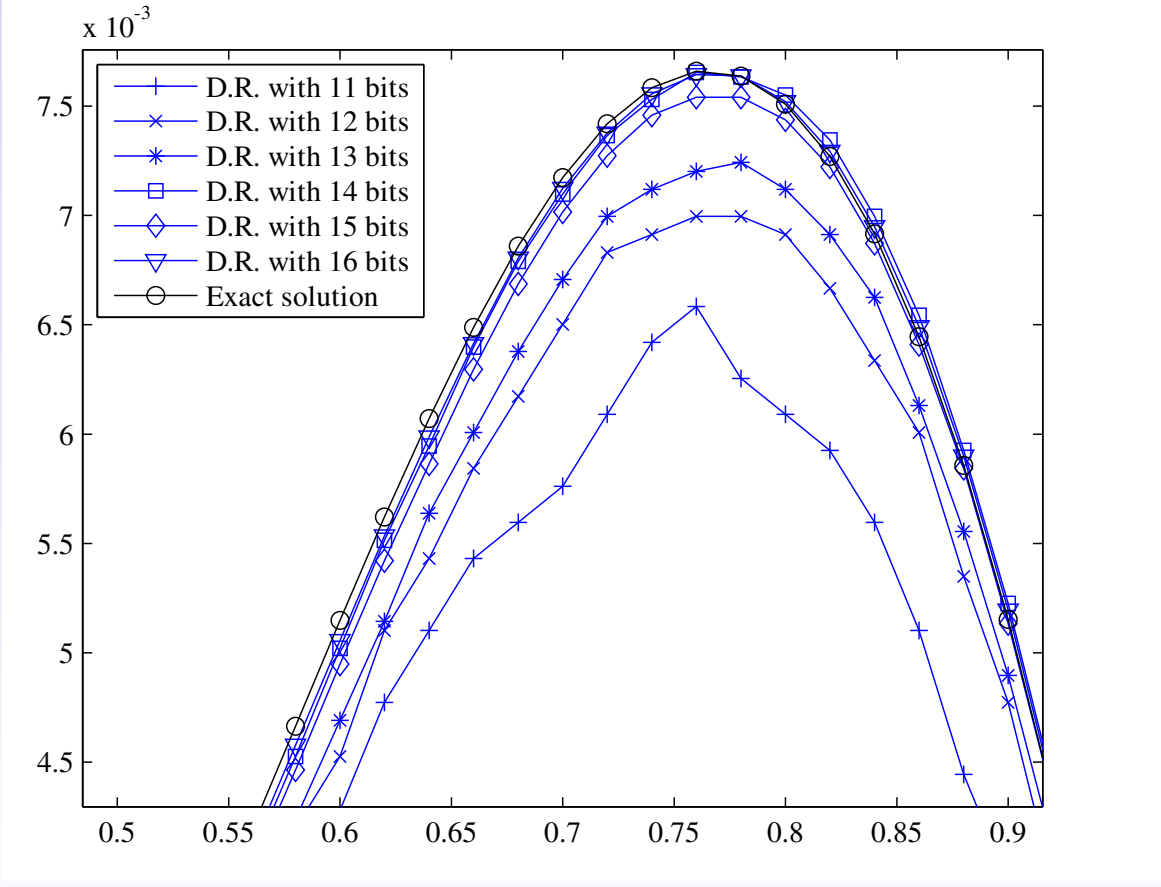


Scaling and quantitization

- General Purpose Processor code use **Floating point type**.
- FPGA by default use **Integer representation**.

Preliminary study:

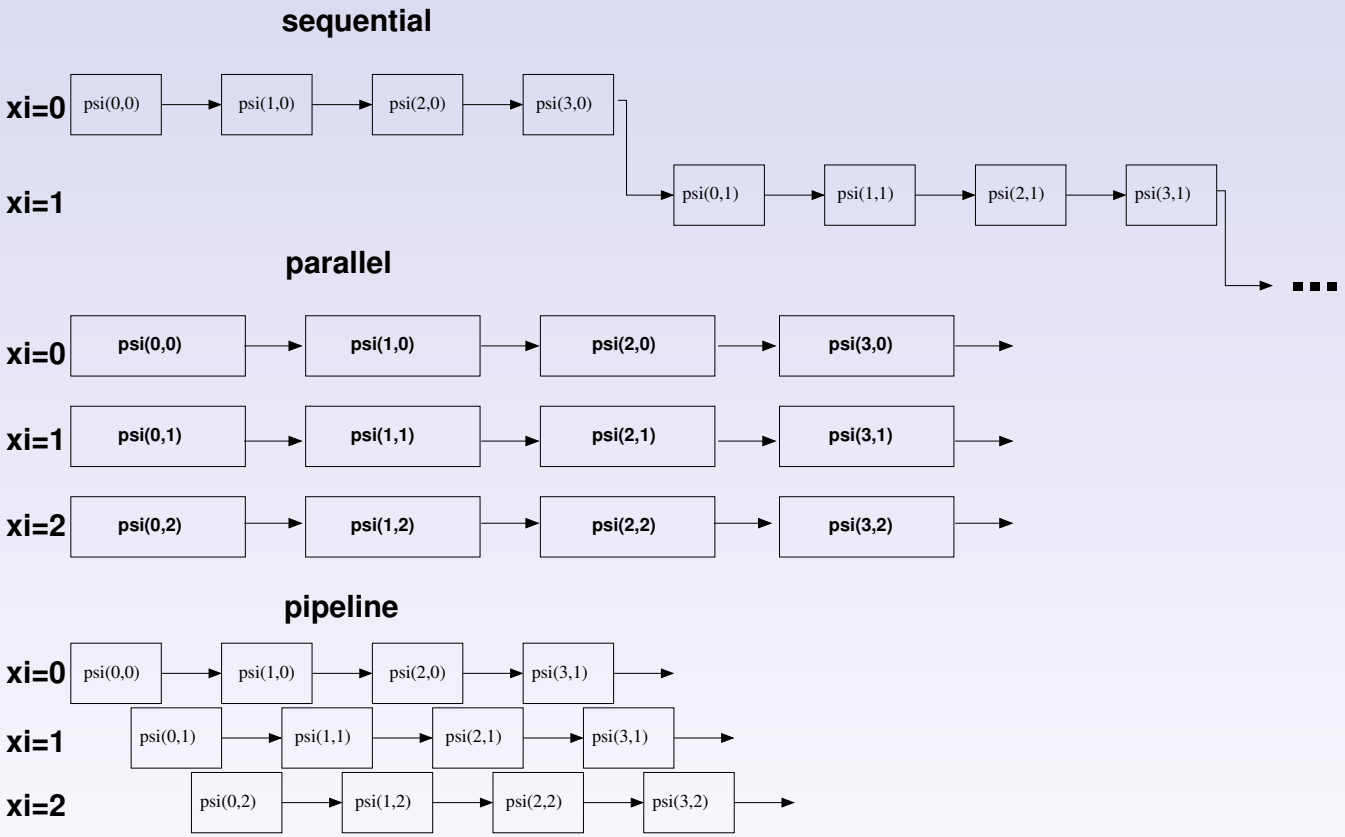
- Data: 9 bits.
- Results: 10 bits.



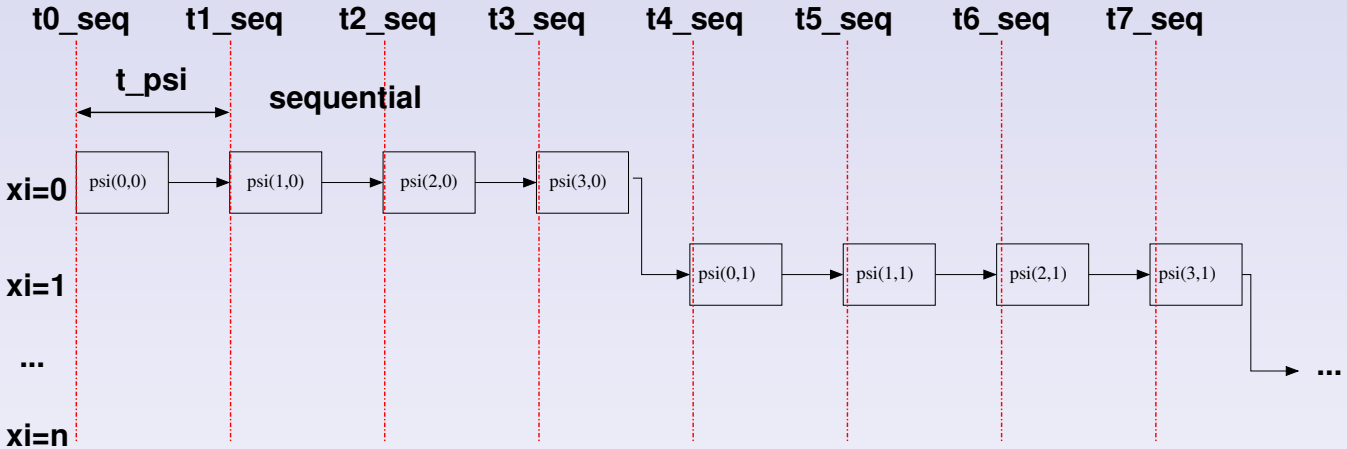
Comparison of accuracy comparison with respect to quantization

Possible implementation solutions

This is not exhaustive.

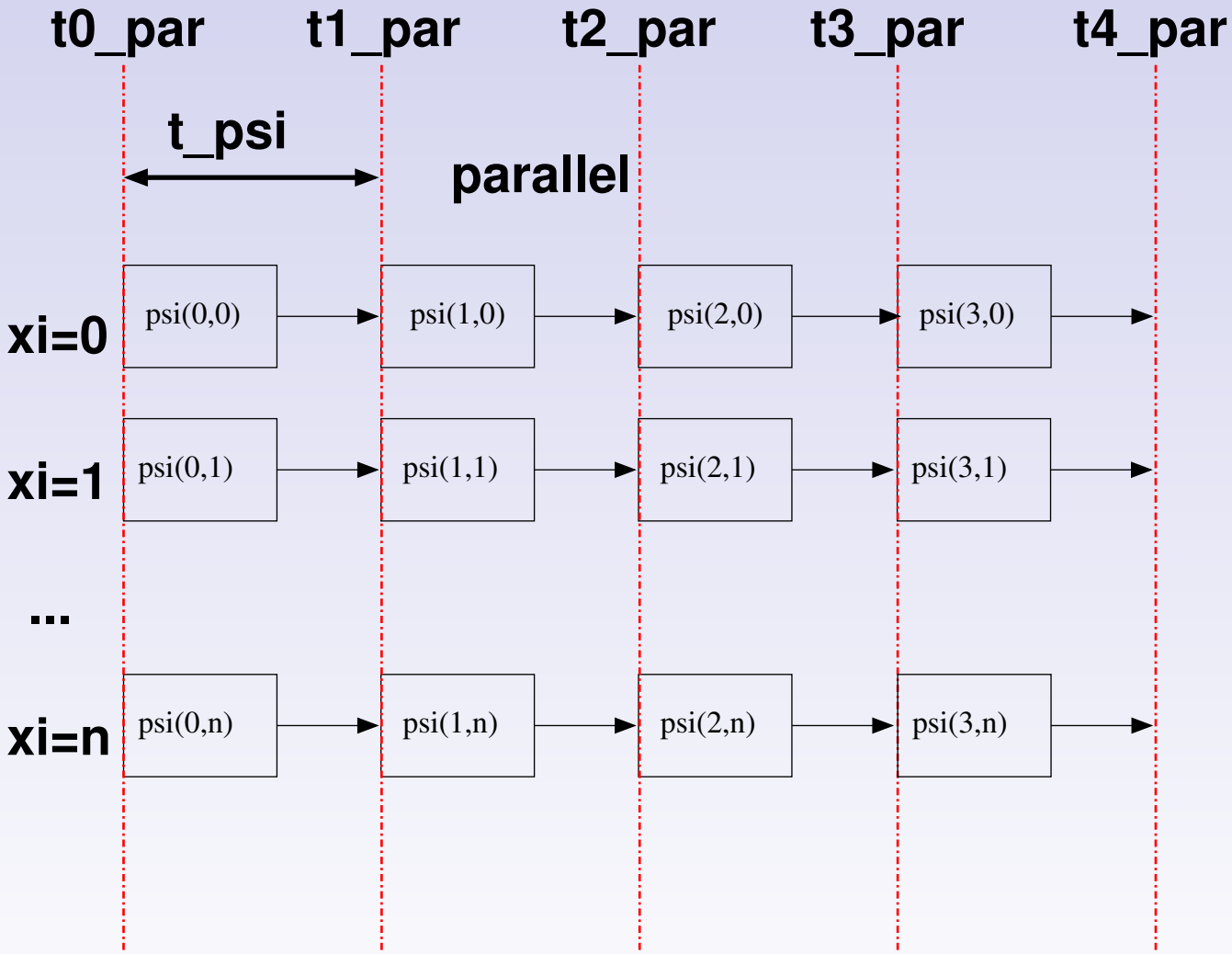


Sequential



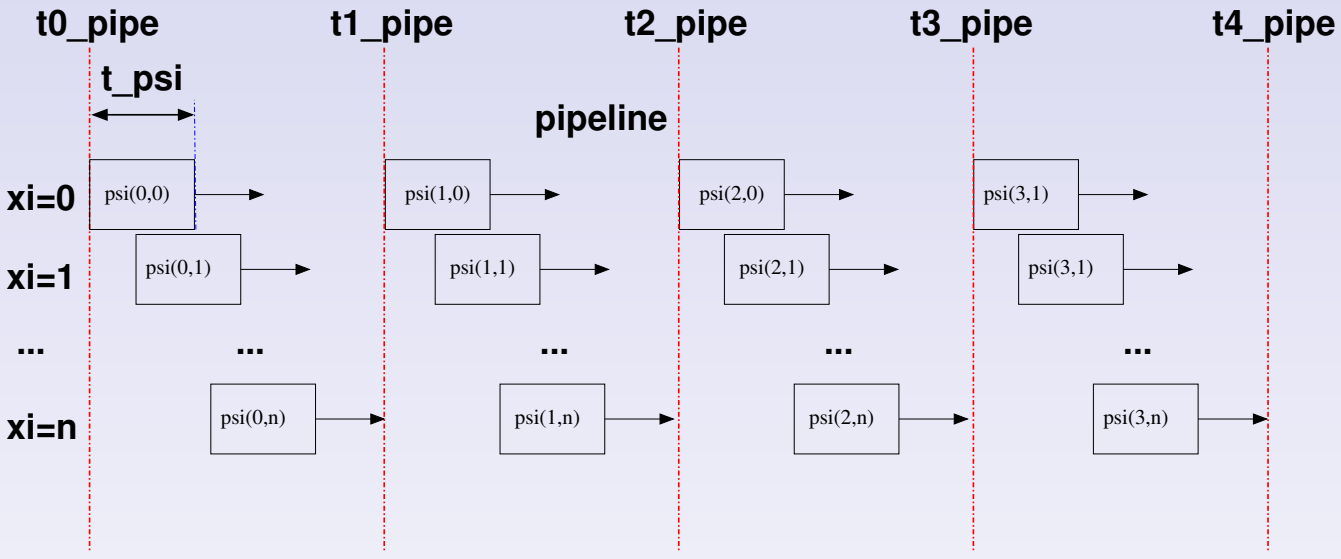
$$t_{tot} = t_{psi} * M * N$$

Parallel



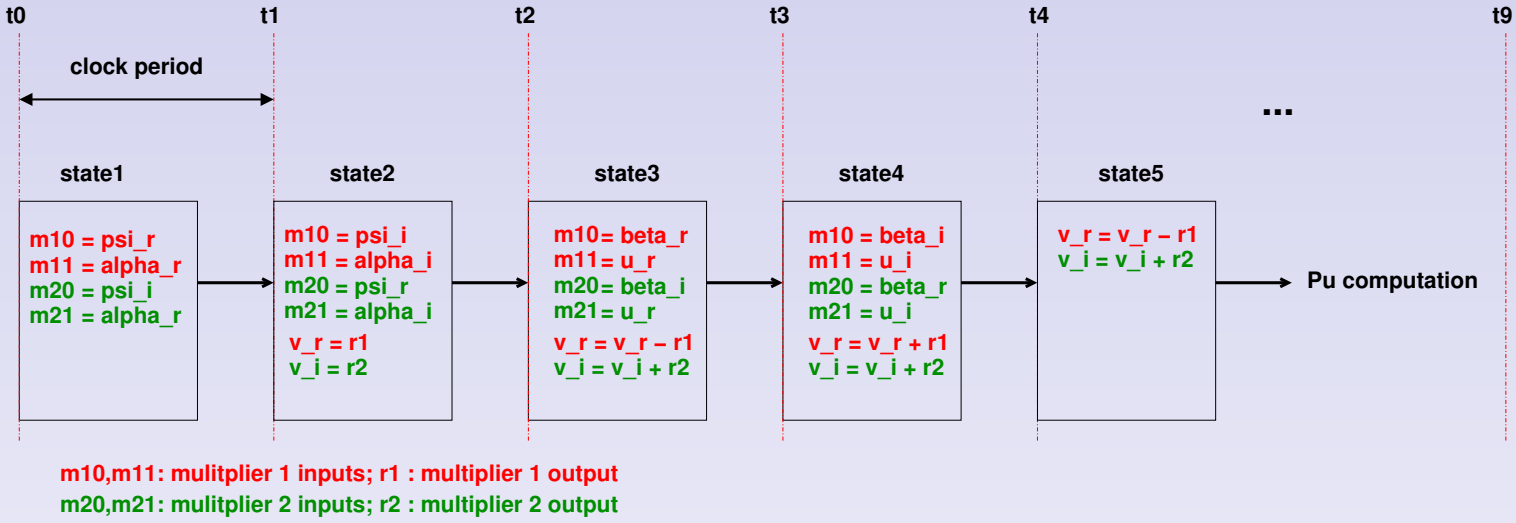
$$t_{tot} = t_{psi} * M$$

Pipeline



$$t_{tot} = t_{psi} + t_{clock} * M * N$$

Parallel



Parallel processing implies:

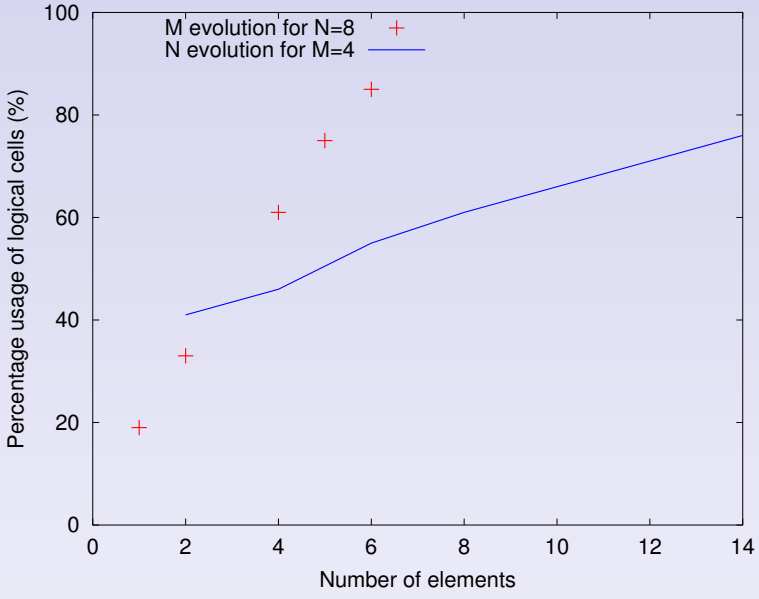
- Distribution of 16 multipliers
- A finite state machine

Implementation is split into two entities:

- The first manages the finite state machine evolution and drive RAM blocks
- The second manages assignment and computation by states.

⇒ Reduce logical dupplication.

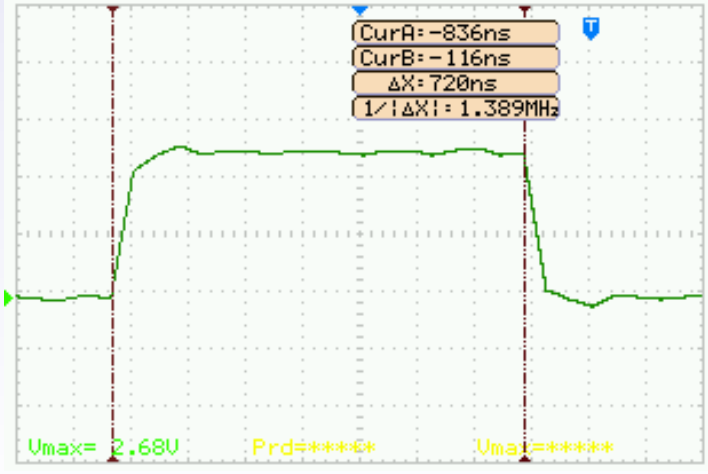
Parallel implementation



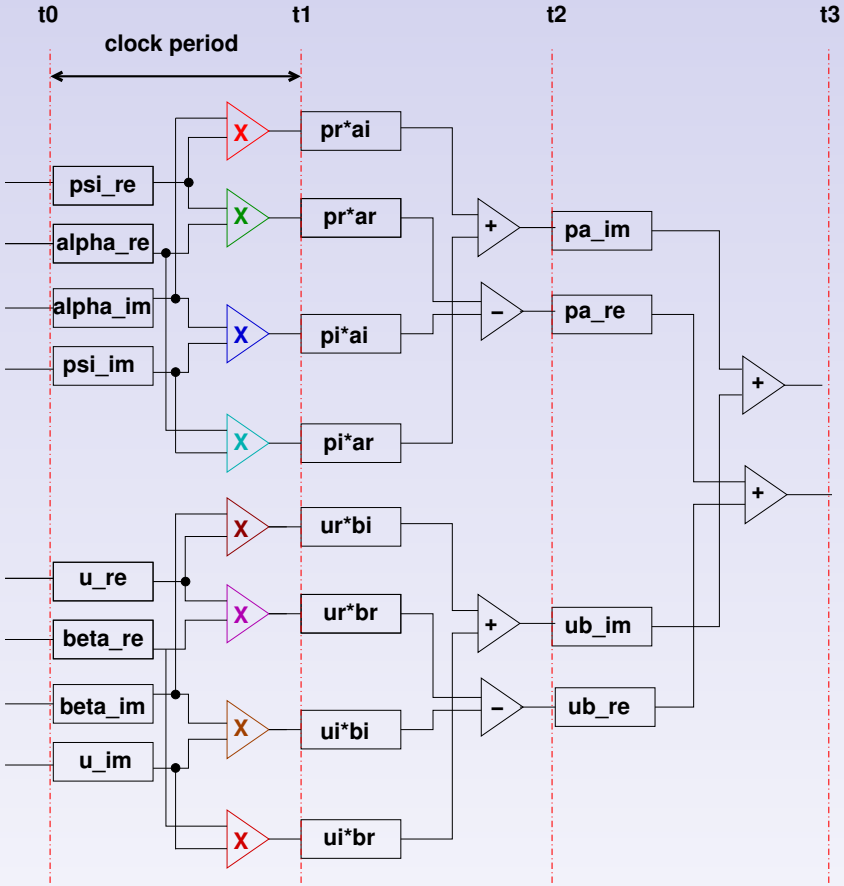
Resource limits:

- $N = 8$ and $M = 6$ (85%)
- or $N = 14$ and $M = 4$ (76%)

Time consumption:
$720ns$ ($N = 4$ and $M = 8$)



Pipeline



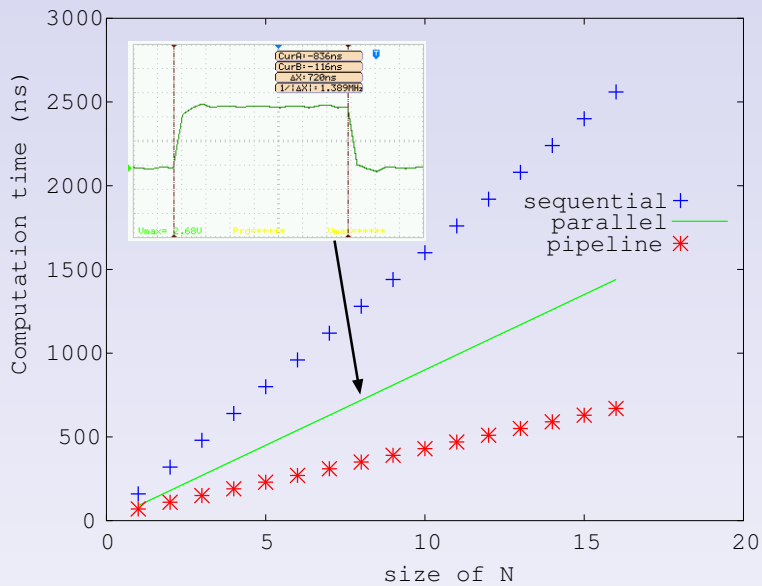
- Implementation not fully validated,
- Using RAM Blocks allows larger data encoding size. $\Rightarrow$ gain in computation precision.

Time consumption estimation

for $M = 4$ and $N = 8$.

FPGA: 100 MHz, PC: 1.6 GHz.

- PC: $\approx$ **60000 ns**
- FPGA:
  - parallel: $720ns$ (Exp.)
  - pipeline: $359ns$ (Part. Exp.)
  - sequential: $1280ns$ (Theor.)
- Ratio: **83 to 167 fold improvement**,
- bandwith: **100Mb/s**.

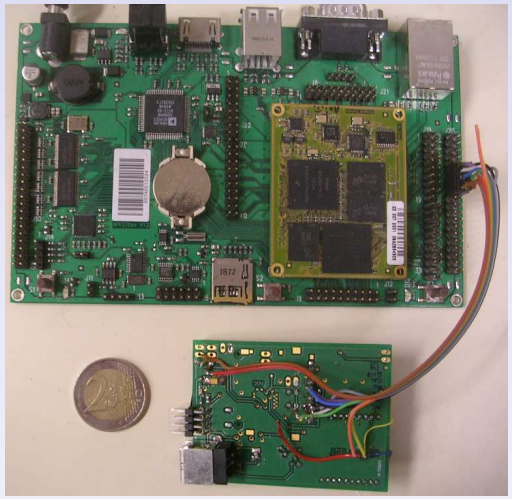


Full task computation time for the three solutions

Computation for a step (t_{psi}) needs $90ns$ for parallel implementation, and $40ns$ for pipeline & sequential implementations

Conclusion and perspectives

- Parallel solution implemented on FPGA. Time computation reduction up to a factor 160.
- Pipeline solution is being validated.



- Today inputs are not from an real device.
- Next Step: embedding into a real physical system
- Extension for an application to arrays of AFM.