Formal Modeling of Run-Time Reconfigurable SoCs for Fault Tolerance Avionics Applications

Daniel M. Muñoz[1] Gilmar S. Beserra[1] Ingemar Söderquist[2]
Ingo Sander[3]

[1]Electronics Engineering Program, Faculty of Gama
University of Brasilia, Brasilia, DF, Brazil

[2]Saab AB, Linköping, Sweden

[3]School of Information and Communication Technology
KTH Royal Institute of Technology, Stockholm, Sweden

Aerospace Technology Congress 2016

Research Question

In order to deal with the increasing complexity of modern safety-critical embedded systems and SoCs, it is important to know:

- → How modeling methodologies can enable the designer to express the behavior of heterogeneous systems at a high level of abstraction?
- → How high level modeling methodologies can be used to design fault-tolerant solutions?
- → How run-time reconfiguration can be integrated to fault-tolerant strategies in a high level of abstraction?
- → How to automate this design process?

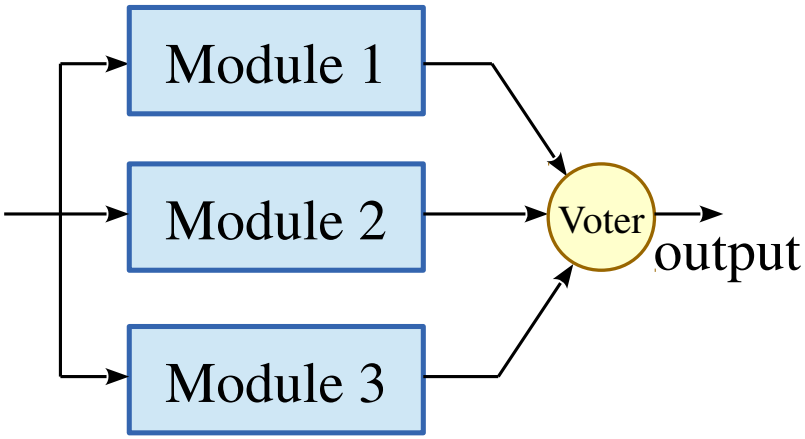
Feasible solutions should naturally lead to:

- → Faithfully generate self-adaptive fault-tolerant implementations of heterogeneous systems in software and digital hardware.

Hardware Redundancy

Triple Modular Redundancy (TMR)

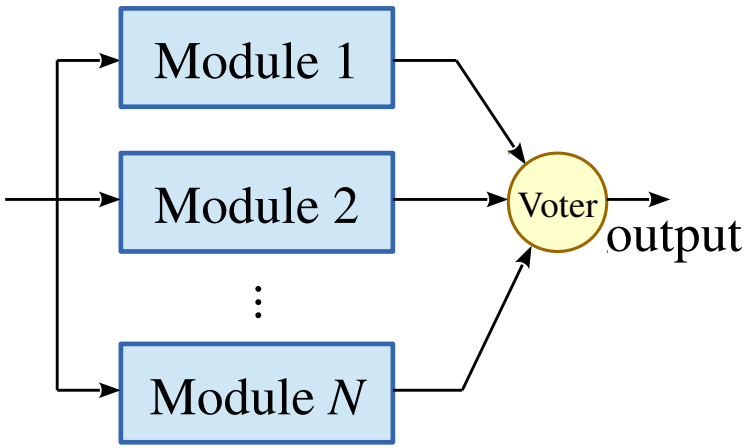
Can detect and mask only one fault.



$$R_{TMR} = 3R^2 - 2R^3 \qquad (1)$$

N Modular Redundancy (NMR)

Can detect and mask $\lfloor N/2 \rfloor$ faults.



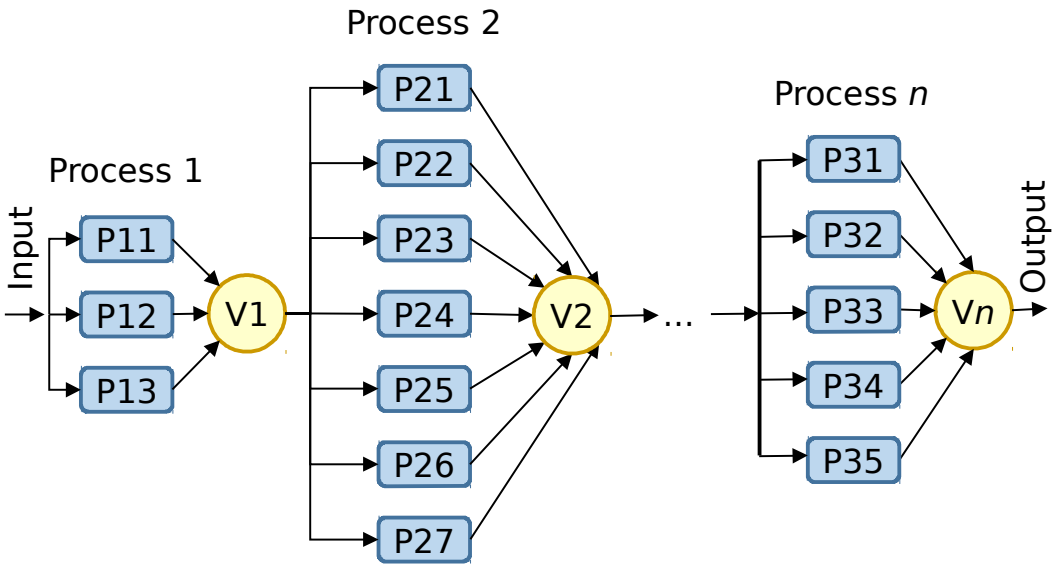
$$R_{NRM} = \sum_{i=n+1}^{2n+1} \binom{2n+1}{i} R^i (1-R)^{2n+1-i} \qquad (2)$$

Redundancy Allocation

Redundant modules must be allocated in a proper way.

Multiple Stage Redundancy

The replication takes place at component level. For independent component failures the multiple stage yields a higher redundancy than high-level redundancy.



ForSyDe (Formal System Design)

ForSyDe is a system design methodology that models systems at a high-level of abstraction. It is based on:

- Theory of models of computations (MoCs)
- Functional programming paradigm

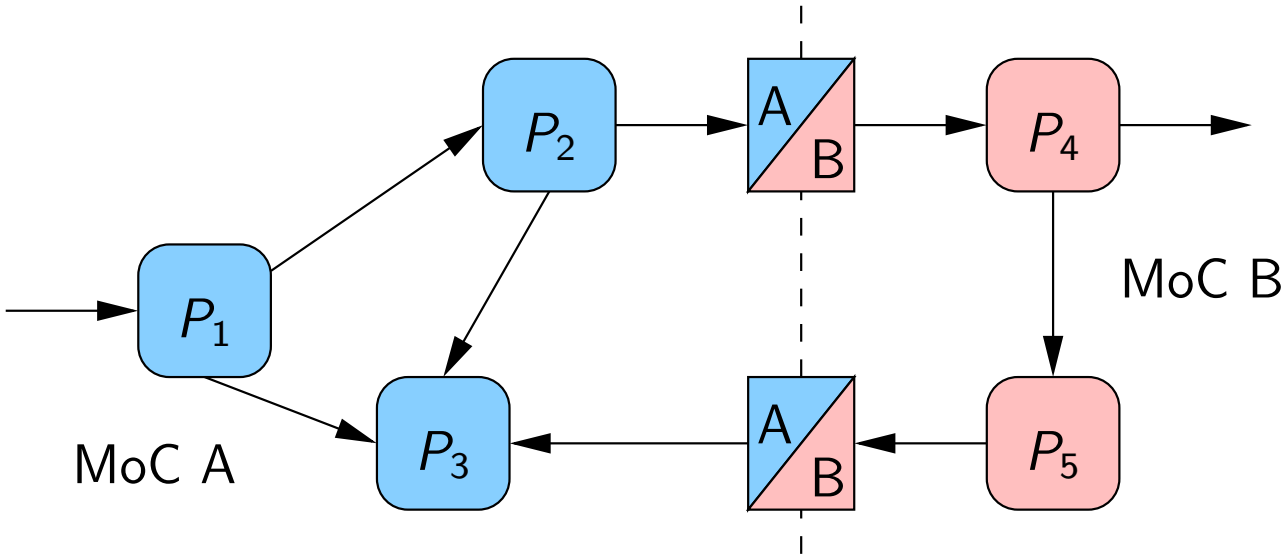
This approach allows for:

- Modeling Heterogeneous embedded systems
- Composition of heterogeneous models
- Modeling adaptive and reconfigurable systems at high-level
- Formal analysis and synthesis techniques by well-defined transformations

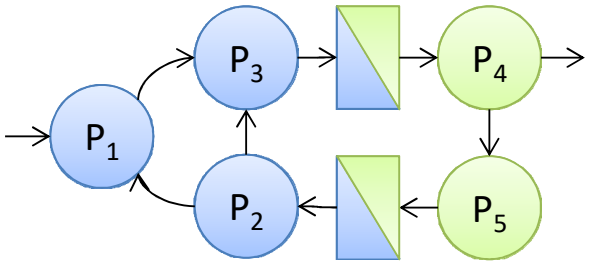
ForSyDe (Formal System Design)

System Model

- A system is modeled as concurrent process network
- Processes belonging to different models of computation communicate via MoC interfaces
- ForSyDe libraries in Haskell and SystemC to support the designer to create a formal model and exist for several MoCs
  - e.g. synchronous MoC, continuous time MoC, SDF MoC

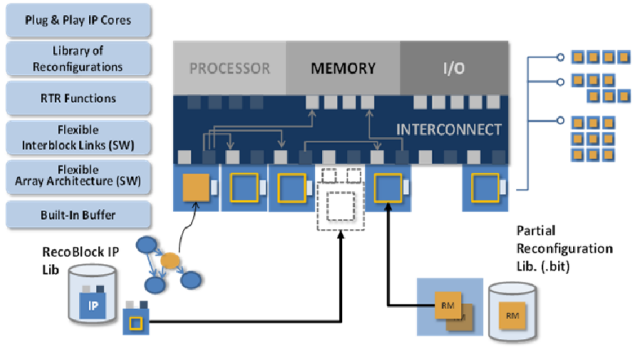


General Adaptive Design Methodology



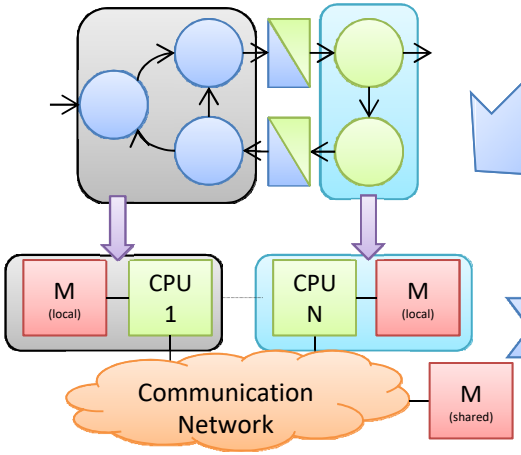
Model of Adaptive System
- Formal Base (MoCs)
- Executable
- Analyzable

Design Constraints
- Real-Time
- Power Efficiency
- Fault-Tolerance
- Robustness
- Safety
- ...



RTR Target Architectue
- Multiprocessor
- RTR Areas + Customizable HW
- Predictable Performance?

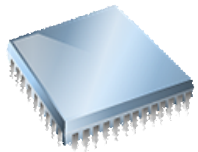
Mapping and Schedule



Design Space Exploration
- Analytical
- Simulation-based (VPs)

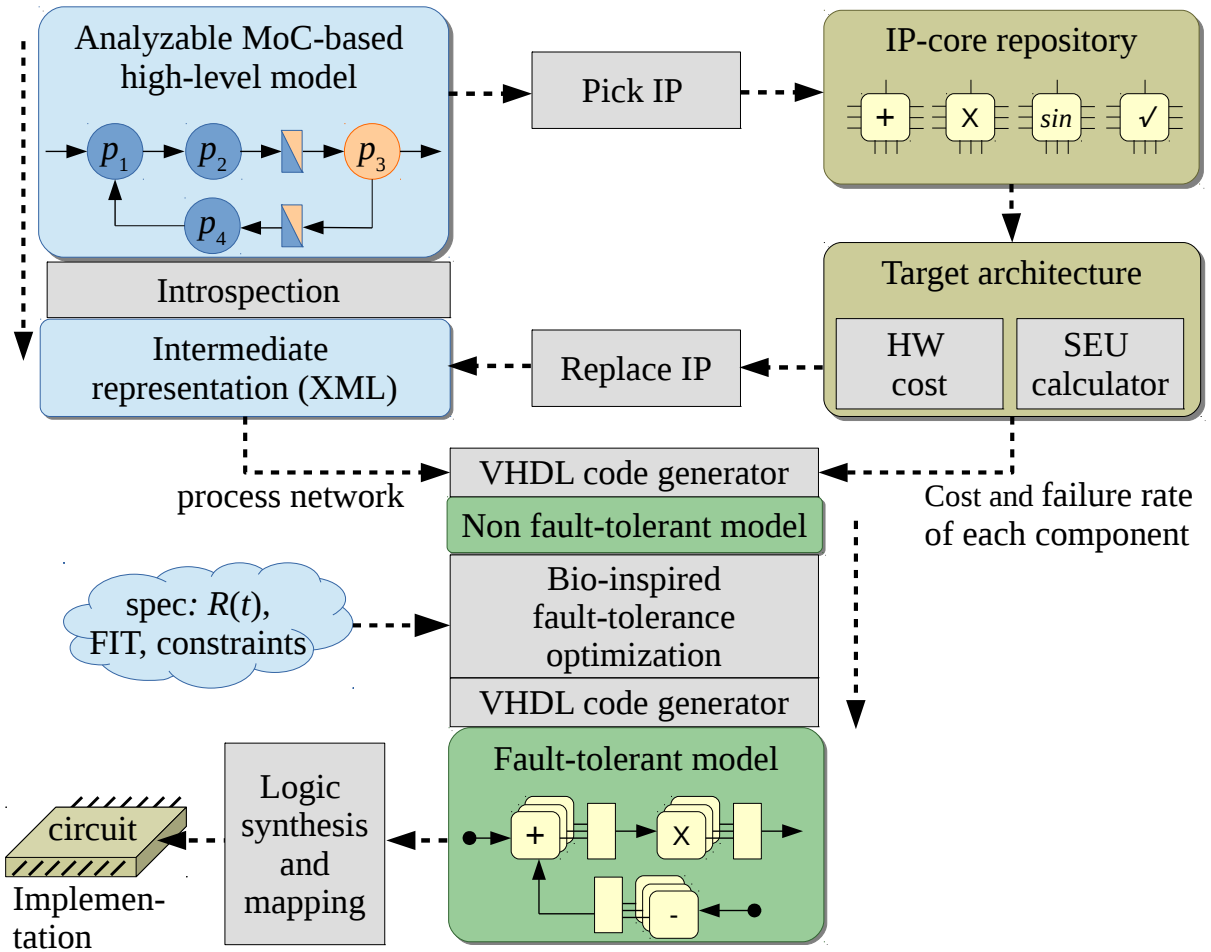
Objective:
Find a mapping that satisfies design constraints

Synthesis and Compilation



Implementation
- Customized HW
- Efficient SW

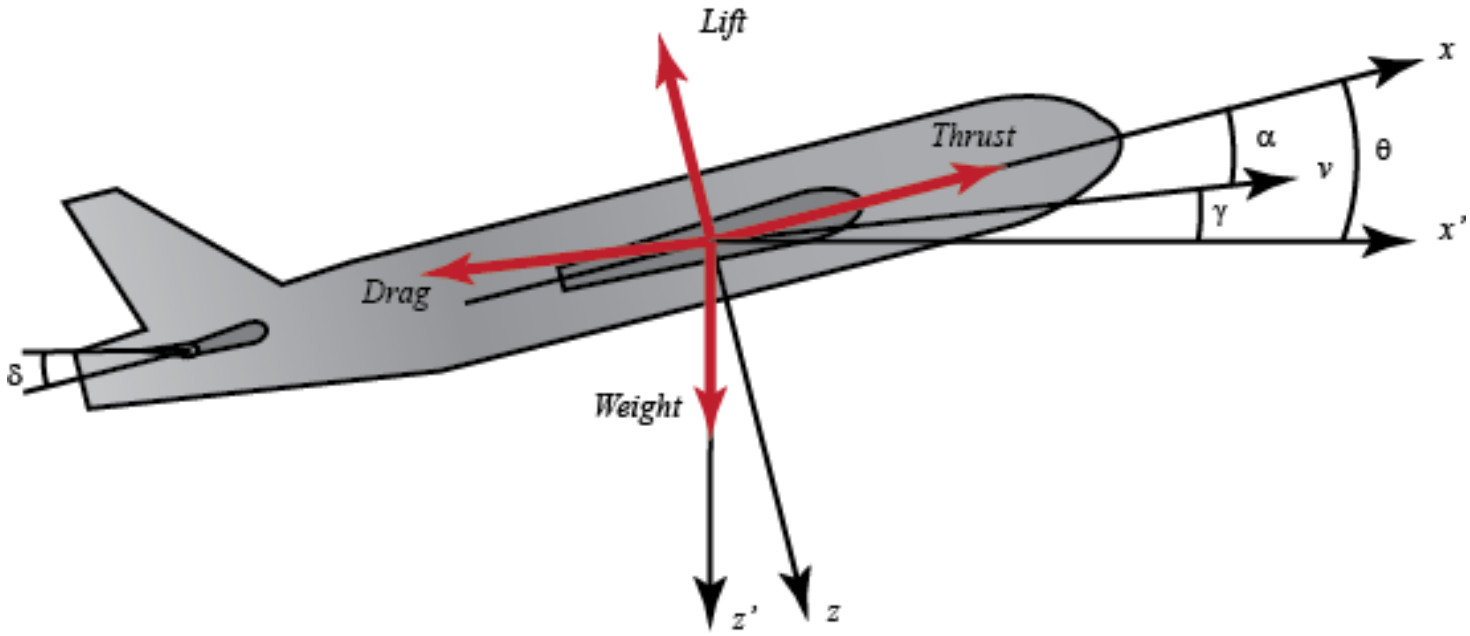
Proposed FT Design Methodology



Based on the semi-formal refinement-by-replacement methodology. Blue rectangles represent high-level models and green ones the fault-tolerant models. Gray boxes are activities for design transformations or characterization.

Case Study: Pitch Control System

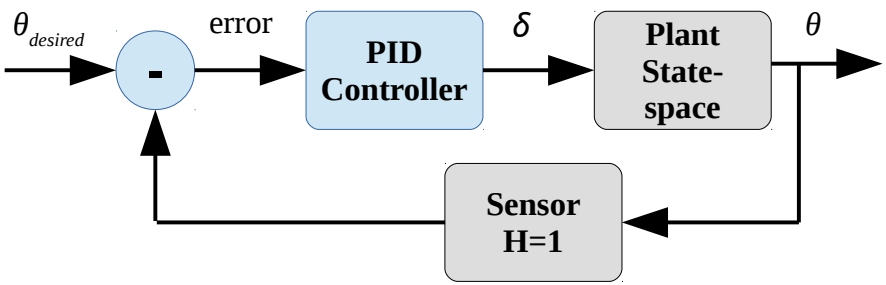
In this case study the objective is to design a fault-tolerant autopilot that controls the pitch angle θ of an aircraft.

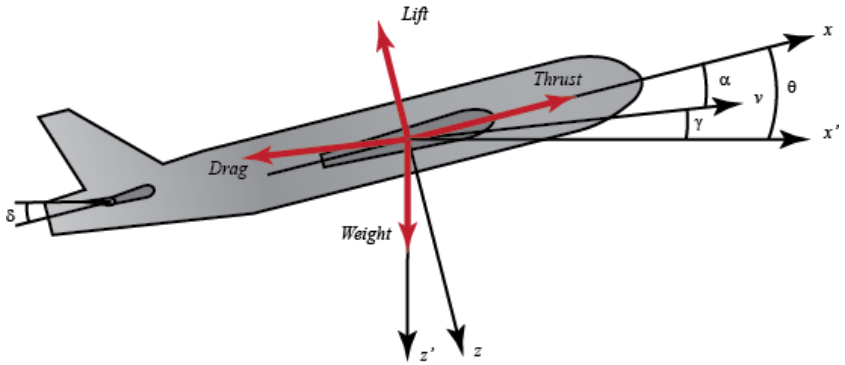


Basic coordinate axes and forces acting on an aircraft. The input is the elevator deflection angle δ and the output is the pitch angle θ of the aircraft. [1]

[1] Case study obtained from *Control Tutorials for Matlab and Simulink*, http://ctms.engin.umich.edu/CTMS/

Case Study: Pitch Control System





- Loop control for modeling the autopilot that controls the pitch of the aircraft. Gray boxes represent the aircraft. Blue boxes will be implemented in a digital circuit with fault-tolerance capabilities.
- A fault-tolerant model of the Proportional Integral Derivative (PID) controller must be automatically obtained and tested on a Zynq FPGA device.

Case Study: Plant Model

Assuming simplifications that the aircraft is in steady-cruise at constant altitude and velocity and that equations governing the motion are decoupled and linearized.

Pitch Formulation for a Boeing's aircraft

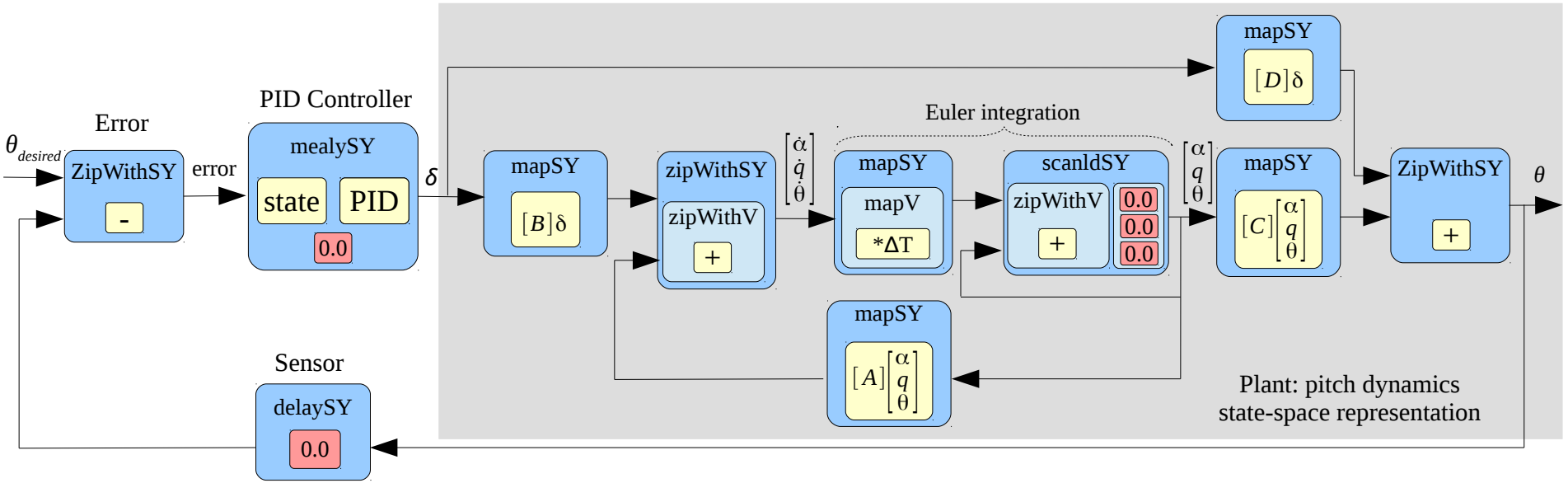
$$\begin{aligned} \dot{\alpha} &= -0.313\alpha + 56.7q + 0.232\delta \\ \dot{q} &= -0.0139\alpha - 0.426q + 0.0203\delta \\ \dot{\theta} &= 56.7q \end{aligned} \tag{3}$$

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \overbrace{\begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix}}^{\text{A}} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \overbrace{\begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix}}^{\text{B}} [\delta] \tag{4}$$

$$y = \underbrace{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}}_{\text{C}} \begin{bmatrix} \alpha \\ q \\ \theta \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{\text{D}} [\delta] \tag{5}$$

1) ForSyDe Implementation of the Overall System

ForSyDe implementation using *process constructors*.

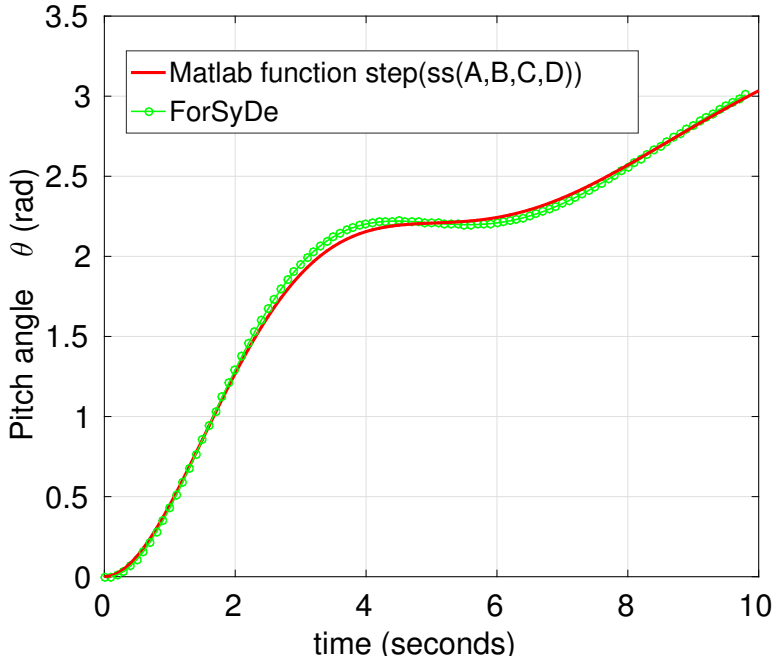


The *sensor process* has a delay to break the feedback loop. The Euler's method was applied for the state integration. The PID controller was implemented using the discrete-time formulation (9).

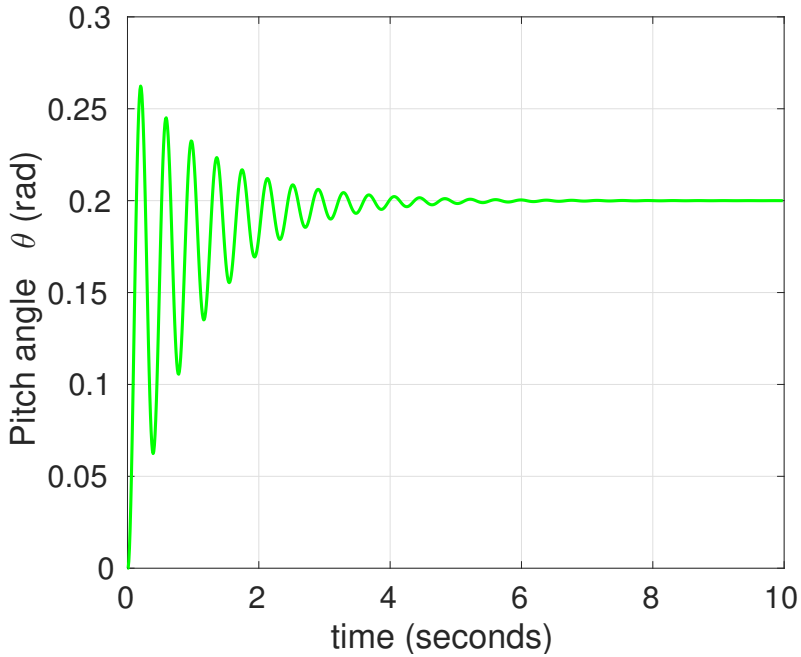
$$\delta[n] = K_p e[n] + K_i(e[n-1] + e[n]) + K_d(e[n-1] - e[n]) \qquad (6)$$

2) Simulation Results of the High-level Implementation

Firstly a comparison of the state-space representation of the plant between ForSyDe and Matlab was performed. The differences are because Matlab makes use of high-order integration methods whereas in ForSyDe we have implemented the first order Euler's method.



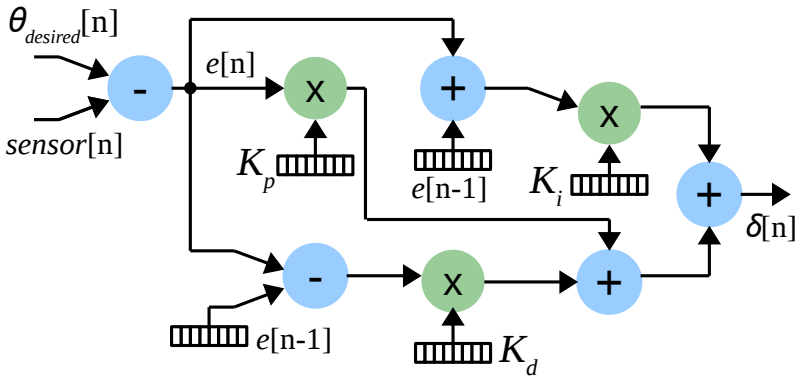
Simulation of the plant: Control signal δ was set to 1.0 during 10 seconds.



Overall system simulation: Set-point was set to 0.2 rad during 10 seconds.

3) Process Network of the PID Controller

- For VHDL code generation the *Error* and *PID Controller* processes were implemented as a process network in ForSyDe-SystemC.



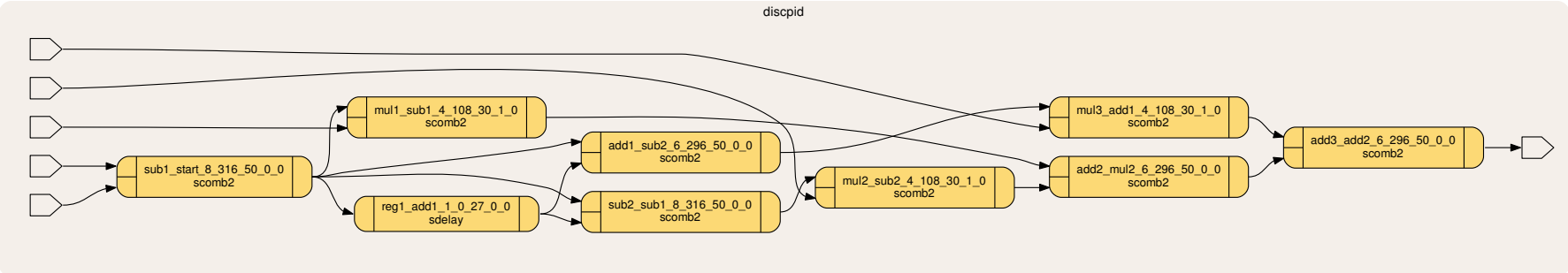
$$e[n] = \theta_{desired}[n] - sensor[n] \quad (7)$$

$$\delta[n] = K_p e[n] + K_i(e[n-1] + e[n]) + K_d(e[n-1] - e[n]) \quad (8)$$

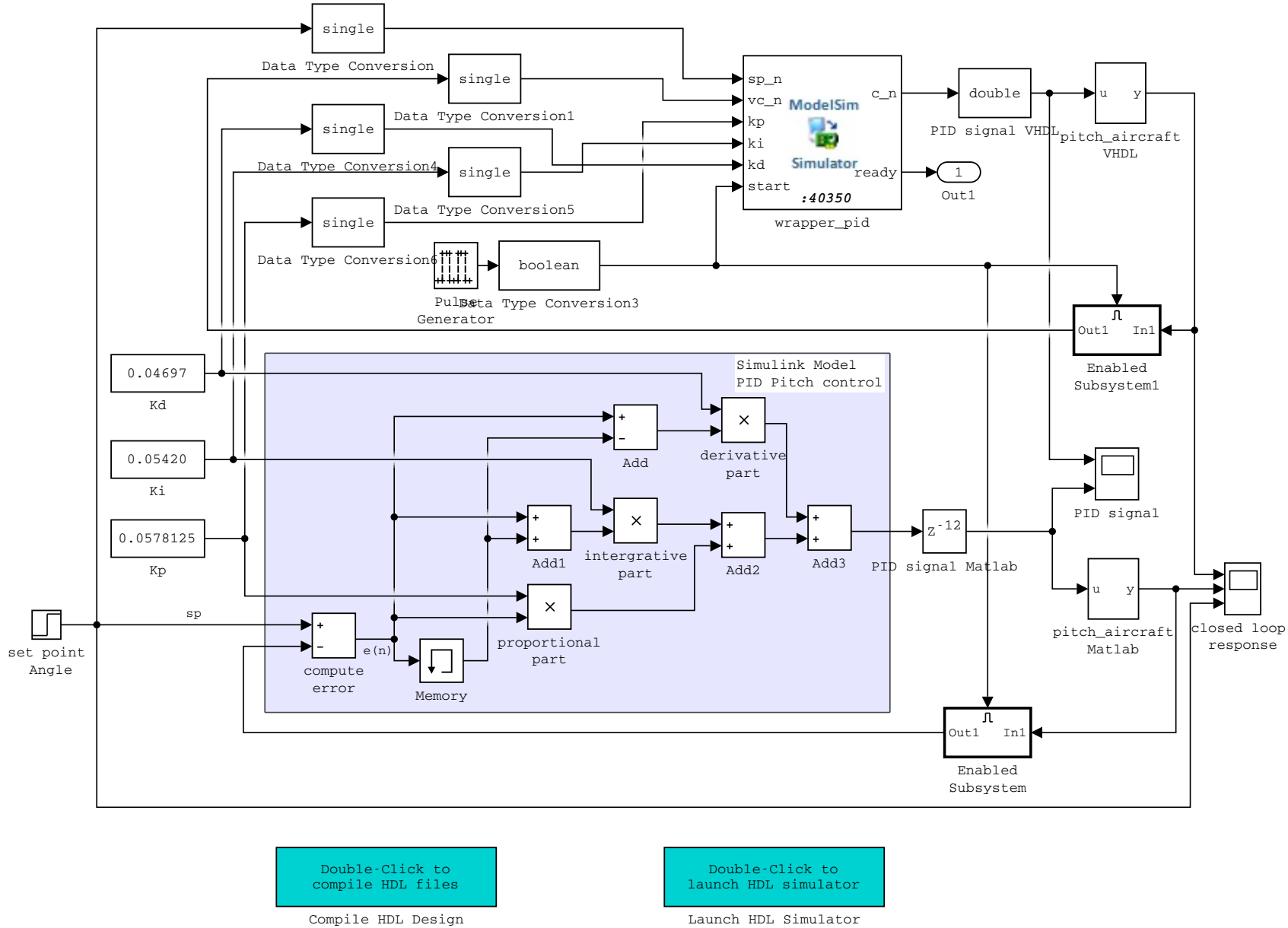
- Three IP-cores were identified: addition, subtraction and multiplication. A custom precision floating-point arithmetics was used (1 signal bit, 8 bits for exponent and 18 bits for mantissa words).
- Previously characterized for a Xilinx Zynq FPGA in terms of hardware cost and failure rate in FIT (data from *Xilinx Reliability Report*).
- More accurate FIT estimations should considers latitude, altitude and mission time of the application.

4) VHDL Code Generation of the PID Controller

- The refinement-by-replacement methodology was applied in order to replace each process in high-level by its respective IP-core.
- It allows the low-level details regarding hardware cost and failure rate of each process to be available in the high-level model.
- The system model as a dot graph was automatically generated by introspection from ForSyDe-SystemC.
- This intermediate representation was read in Matlab and the developed VHDL code generator tool automatically provides the VHDL model with components instantiations and connections.

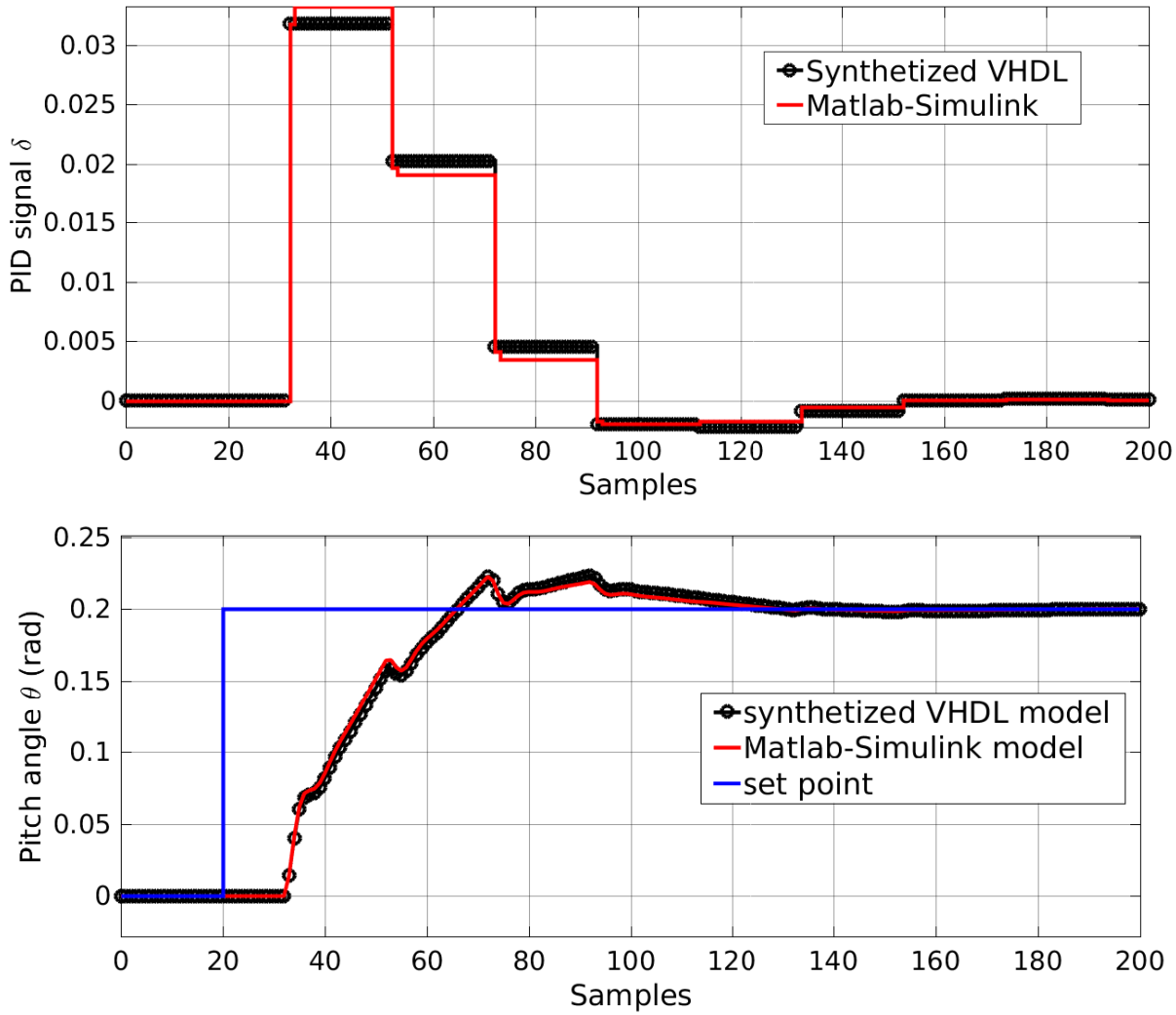


5) Co-simulation Environment



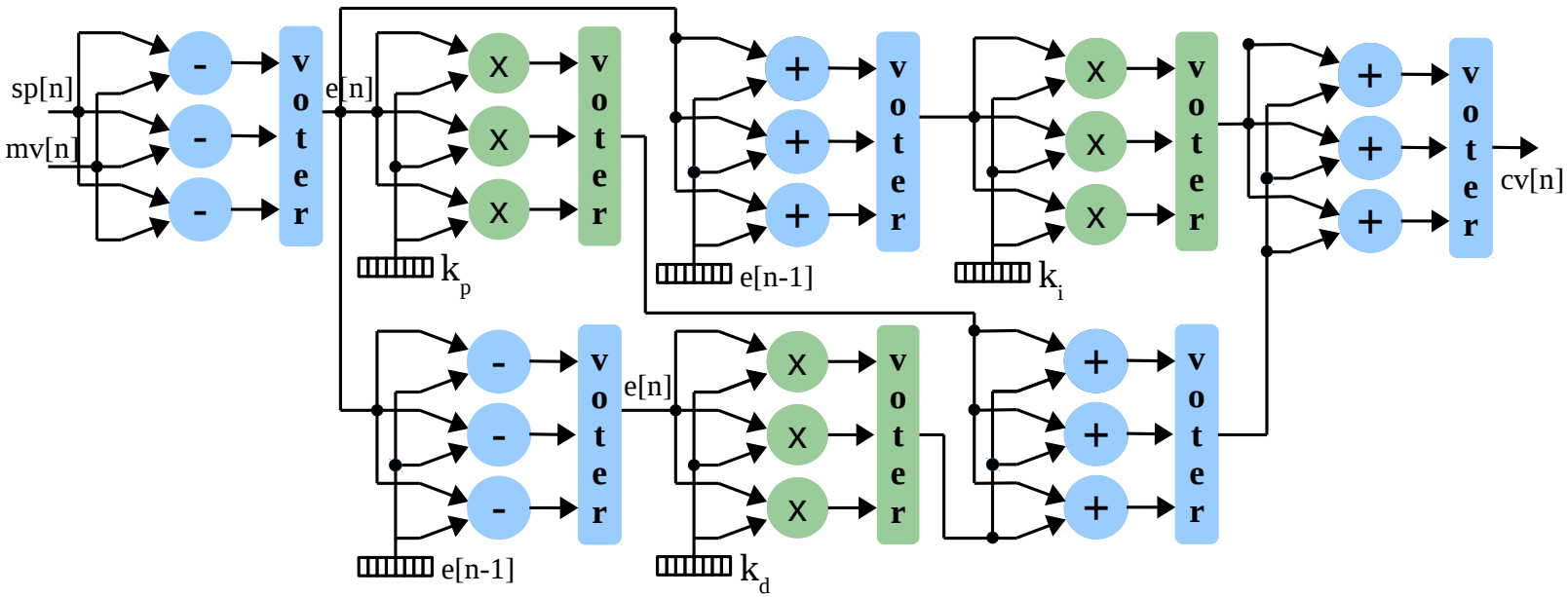
5) Simulation Results of the Obtained VHDL Model

HW/SW comparison. Top: Control signal δ. Bottom: Pitch angle θ



6) Fault-Tolerant Solution using TMR

- After evaluating the correct behavior of the non fault-tolerant solution, a fault-tolerant model of the PID controller for the pitch of an aircraft was **manually developed** using the multiple-stage TMR scheme.
- Each component of the process network was triplicated and a majority voter was used.



6) Characterization of the TMR Solution

FIT estimation and hardware cost for the non fault-tolerant and the TMR fault-tolerant models

Architecture	FIT $\times 10^9$	LUTs (53200)	FFs (106400)	DSP48E (220)	BRAM (140)
non-fault-tolerant	46.99	1856 (3.5%)	367 (0.34%)	3 (1.4%)	0 (0.0%)
Majority Voter	0.0	27 (0.05%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
TMR fault-tolerant	7.367	6795 (12.8%)	1101 (1.03%)	9 (4.1%)	0 (0.0%)

The TMR scheme improved the system reliability from 0.6399 to 0.9324. FIT values were estimated using the TMR reliability equation assuming perfect voters.

Improving the TMR results

Better results can be achieved using the optimization tool of the design flow. This tool automatically provides the VHDL code of the fault-tolerant solution based on the NMR scheme.

7) Characterization of the NMR Solution

FIT estimation and hardware cost for the non fault-tolerant and the TMR and NMR fault-tolerant models

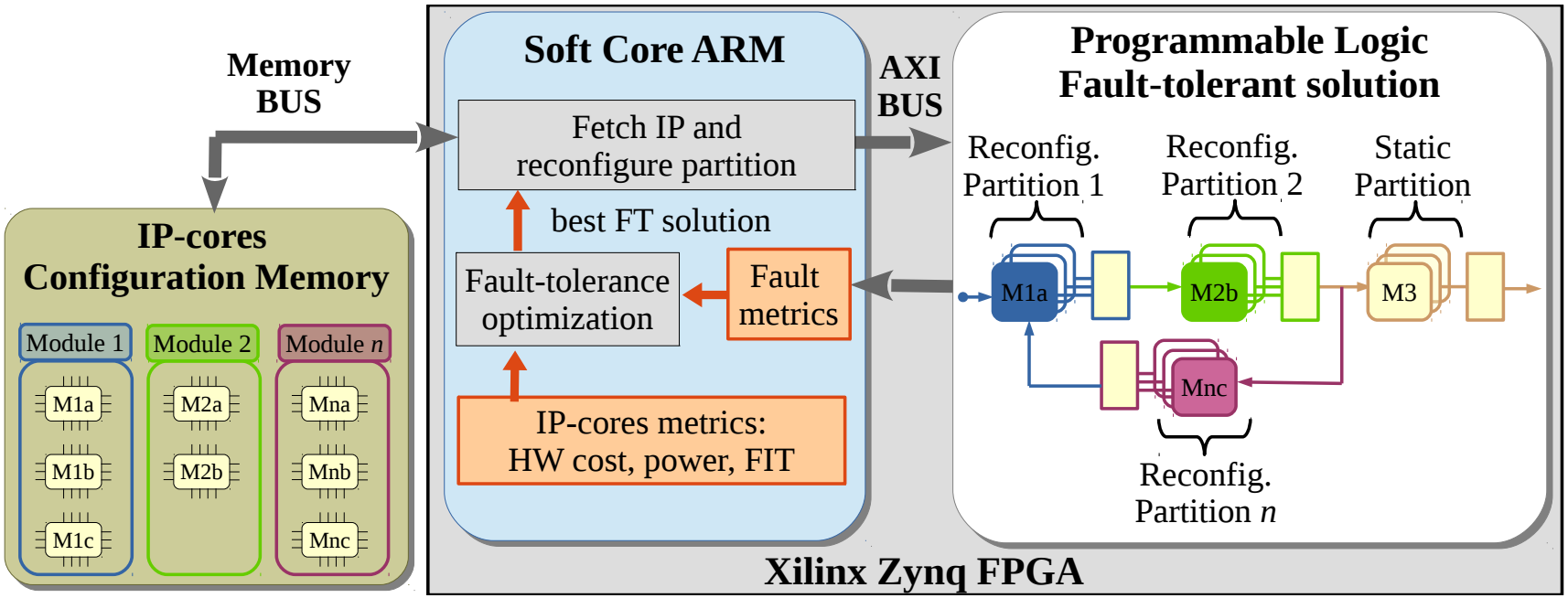
Architecture	FIT $\times 10^9$	LUTs (53200)	FFs (106400)	DSP48E (220)	BRAM (140)
non-fault-tolerant	46.99	1856 (3.5%)	367 (0.34%)	3 (1.4%)	0 (0.0%)
Majority Voter	0.0	27 (0.05%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
TMR fault-tolerant	7.367	6795 (12.8%)	1101 (1.03%)	9 (4.1%)	0 (0.0%)
NMR fault-tolerant	0.423	14958 (28.1%)	2335 (2.2%)	15 (6.8%)	0 (0.0%)

- All the architectures fit in the FPGA device.
- Achieved NMR scheme using 7, 5, 7 replicas for the subtraction, multiplication and addition components, respectively.
- Reliability of the NMR system equal to 0.9960.

Summary

- → This work has presented a fault-tolerance design methodology from abstract models.
- → *Innovative aspect:* the use of a formal system-level design based on models of computations that can be integrated with low-level dependability information through the use of IP-cores.
- → Bio-inspired algorithms and an automatic VHDL code generator tool were integrated in the design flow in order to optimize the fault-tolerant solution in terms of reliability and hardware cost.
- → The pitch control system of an aircraft was used as case study.
- → The achieved NMR solution was mapped on a FPGA device.
- → The NMR solution improved the reliability of the system from 0.6399 to 0.9960.

Future Works



→ Run-time reconfigurable systems will be modeled in high-level. It enables to develop self-adaptive fault-tolerant solutions using heterogeneous architectures.

→ Timing and hybrid redundancy solutions should be included.

→ Execution time precision and power consumption criteria as a multi-objective fault-tolerance optimization problem.

Acknowledgment

This work was developed with the support of CNPq, National Council of Scientific and Technological Development of Brazil, of CISB, Swedish-Brazilian Research and Innovation Centre, and of Saab AB.

Thanks for your attention

Do you have questions?

Contact:
Daniel M. Muñoz
e-mail: damuz@unb.br